VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Realizace síťové strategické hry

Realization of a network strategic game

2010 Lubomír Březina

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. května 2010 .....

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce, panu Ing. Marku Běhálkovi Ph.D. a všem kolegům, kteří mi poskytli své rady, protože bez nich by tato práce nevznikla.

Abstrakt

Bakalářská práce se zabývá problémy spojenými s vývojem počítačové hry. Je vypracována jako část spolupráce na společném projektu - základu počítačové hry. Spolupracující student Lumír Janošek se zabývá problematikou související s vykreslováním grafiky hry a optimalizacemi vykreslování scény. Má část se zabývá problémem řízení agentů (postav ve hře) a to jak pohyb po herní mapě, tak také jejich chováním. Dalším problémem, kterým se tato bakalářská práce zabývá je síťová komunikace hry v počítačové síti LAN, kde se zabývám synchronizací a přenosem aktuálních informací o průběhu hry. V poslední řadě se zaobírám samotným herním systémem, kde stanovuji podmínky chování a interakci hry s uživatelem-hráčem. Výsledkem celé práce je technologické demo - spustitelný program, nebo sada programů, které předvádí funkčnost implementovaných dílčích částí.

Klíčová slova: umělá inteligence, agent, síťová hra

Abstract

The thesis deals with problems concerning the development of a computer game. This is intended to be the basis of a computer game and it is a collective work of two students. Lumír Janošek – the cooperating student deals with problems concerning a scene rendering and scene optimization. My part deals with the artificial intelligence – the movement of game characters on the game map and their behavior. The other problem the thesis deals with is a net communication of the game in LAN computer network where the synchronization and transfer of actual information about the game behavior are examined. In the last part the game system is dealt with and the conditions of the game behavior and the interaction of the user and player are set. The result of the thesis is a technological demo – an executable file or a set of programmes that shows the functionality of implemented fragments.

Keywords: artificial inteligence, agent, network game

Seznam použitých zkratek a symbolů

LAN	–	Anglická zkratka Local Area Network je označení pro lokální počítačovou síť.
AI	–	Anglická zkratka Artificial Inteligence je označení umělé inteligence.
agent	–	Postava řízená umělou inteligencí nebo hráčem.
FPS	–	Anglická zkratka Frame Per Second znamená počet snímků obrazu za jednotku času.
socket	–	Kanál pro přenos dat mezi dvěmi propojeným aplikacemi ve Winsock2.
API	–	Anglická zkratka Application Programming Interface označuje rozhraní pro programování aplikací.
RPG	–	Zkratka Role-playing game znamená hru na hrdiny.
multicast	–	Je způsob přeposílání IP datagramů z jednoho zdroje skupině koncových stanic.
QoS	–	Quality of Service - Zajištění kvality přenosového kanálu pomocí vyhrazování přenosových kanálů
A*Star	–	Varianta Dijkstrova algoritmu s heuristikou pro hledání nejkratší cesty v grafu. Více v literatuře [2]
Dijkstrův algoritmus	–	Dijkstrův algoritmus hledá nejkratší cesty v grafu. Jeho prohledávání je do šířky. Více v literatuře [2]

Obsah

Seznam obrázků

1 Úvod

Počítačové hry jsou velmi významnou součástí rozvoje informačních technologií. Díky potřebě hráčů hrát stále propracovanější herní tituly s realistickou krajinou a chytrou umělou inteligencí, je vyvíjen tlak na výrobce, kteří se snaží poskytnout výkonnější hardware, který bude schopen tyto nároky zvládnout. Odvětví vývoje počítačových her určuje trend moderním technologiím, ověřuje již zažité algoritmy, ale nutí také vytvářet nové. Vývoj počítačových her dnes napomáhá rychlejšímu vývoji technologií. Nejvíce viditelný pokrok je v grafice počítačových her. Ten je dán značným nárůstem výkonu a současným (paralelním) zpracováním výpočtů v plovoucí čárce a neustálým vylepšováním technologií, díky kterým jsou efekty daleko věrohodnější a je možné vykreslit náročné scény s miliony polygonů více než dvacetpětkrát za sekundu. Počítačové hry však nejsou postaveny jen na grafice. Každá počítačová hra je souborem mnoha dílčích částí, které jsou spojeny v jediný komplexní funkční celek.

Téma vývoje technologického dema počítačové hry, pracovně byla tato hra pojmenovaná *Heroes*, jsem si vybral z důvodu mého zájmu o počítačovou zábavu a proto, že bych se v budoucnosti tímto směrem rád ubíral. Mým hlavním zájmem však není grafika, ale logika hry. V této bakalářské práci se budu zabývat jen některými zajímavými částmi, které jsou pro hráče již samozřejmé, ale bez kterých by hra nefungovala. Pro naší budoucí počítačovou hru, která bude kombinací herních titulů typu RPG/strategie shrnu v prvních kapitolách krátce použité technologie a existující řešení problému hledání cesty. Dále krátce specifikaci a poté architekturu hry. Na nich bude navazovat kapitola herního systému, který bude pouze náhradou za pozdější jádro hry. Na herní systém naváže kapitola umělé inteligence, která se bude zabývat problémy hledání cesty a řízením postav AI agentů. V poslední části se zaměřím na problém hraní hry po síti a jeho řešení. Nakonec uvedu možnosti ozvučení a v závěru shrnu dosažené výsledky.

2 Technologie

Při implementaci jsem využil, nebo zvažoval využití těchto technologií a knihoven.

1. programovací jazyk C++ - implementace programu
2. technologie Microsoft DirectX - vykreslování grafiky a ovládání hry
3. Microsoft Windows Sockets 2 - rozhraní pro přenos dat v síti
4. knihovně FMOD - knihovna pro přehrávání zvuků
5. knihovně OpenAl - knihovna pro přehrávání zvuků
6. knihovna ZThread - knihovna pro vlákna

2.1 DirectX

Microsoft DirectX je balík knihoven postavený pro platformu Windows, která zajišťuje jednotnou vrstvu API pro vývoj multimediálních aplikací. Zajišťuje nízko úrovňový přístup k zařízení počítače nezávisle na jeho hardware. DirectX obsahuje rozhraní Direct3D pro 2D a 3D grafiku, DirectInput, API které zajišťuje sběr vstupů od uživatele a to přes všechny možné zařízení od klávesnice, myší, joysticku a dalších ovladačů. DirectInput zároveň zajišťuje zpětnou vazbu pro dosažení reálnějších pocitů ze hry. Další součástí je Windows Game Explorer, který je přidán jako podpora hry pro systémy Windows Vista a Windows 7, kde je možné každé vyvíjené hře přidat popis, nastavit pro koho je hra určena, shrnout hardwarové nároky a přidat obrázek. Systém se postará o začlenění hry do systému Windows a její jednodušší správě. DirectX ještě obsahuje rozhraní od kterých se bude postupně upouštět a jejichž práci přeberou konkurenční produkty. Mezi nimi je API pro síťové hraní DirectPlay, které se doporučuje nahradit standardními Winsock2. Dále pak DirectSound a DirectMusic, které se staraly o zvuk ale konkurenční produkty FMOD a OpenAL již tyto rozhraní překonaly a v poslední řadě to je DirectDraw, pro kreslení 2D grafiky. Více informací najdete na www stránkách viz [5].

2.2 WinSock2

Windows sockets 2, dále jen Winsock2 jsou součástí rozhraní Win32 API, které umožňují programátorům vytvářet internetová, intranetové a ostatní síťové aplikace přenášející aplikační data nezávisle na použitém síťovém protokolu. Winsock2 umožňují přístup k pokročilým síťovým funkcím jako je multicast a QoS. Winsock vychází z návrhu socketů využívaného poprvé v systémech BSD. Rozhraní se postupně rozšířilo na všechny platformy a Winsock2 je jeho Windows implementací. Avšak implementace ve Windows je mírně rozšířena. Základ však zůstává stejný a mohou mezi sebou komunikovat i aplikace z různých operačních systémů. Winsocks2 zvládají v případě potřeby využití více protokolů v jedné aplikaci současně. Více informací se dočtete na stránkách MSDN společnosti Microsoft, viz [6].

2.3 FMOD

FMOD je nejpoužívanější softwarová API pro vytváření a přehrávání interaktivního zvukového doprovodu. FMOD je nejrozšířenější rozhraní pro zvuk používané v herním a průmyslu. Největší výhody API FMOD jsou jednoduché použití, výkonné softwarové jádro pro mixování zvuku a multiplatformní architektura. FMOD je distribuován také pro více různých hardwarových platforem, jako jsou herní konzole a to usnadňuje vytváření konzolových verzi hry bez nutnosti psát program znovu. Pro nekomerční účely je API poskytováno zcela zdarma. Více informací najdete na www stránkách viz [8].

2.4 OpenAL

OpenAL je multiplatformní softwarové API pro prostorové přehrávání zvuku navržené primárně pro počítačové hry a aplikace. Model API je postaven tak, aby bylo možné zvuky umísťovat přímo do 3D scény, aby měl hráč skutečný pocit o směru a intenzitě zvuků. Toto řešení vylepšuje hráčovu prostorovou orientaci ve scéně. Knihovna je také distribuována zdarma. Více informací najdete na www stránkách viz [9].

2.5 ZThread

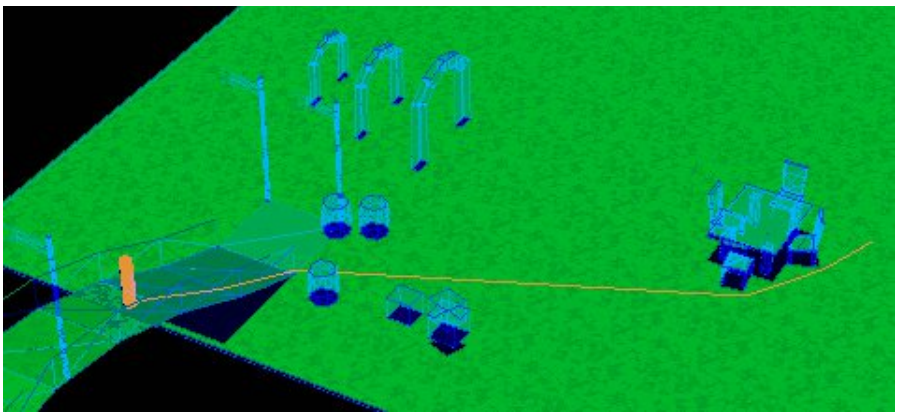
ZThread je multiplatformní dynamická knihovna pro vlákna a synchronizaci pro programovací jazyk C++ distribuovaná pod licencí "The MIT License". Knihovna má otevřené zdrojové kódy. Podporuje zámky a je objektově orientovaná. Knihovna je koncipována pro velmi jednoduché spravování vláken. Podrobné informace a dokumentace se nalézá na stránkách projektu [7].

3 Existující řešení

Projektů zabývajících se hledání cesty je na internetu velké množství. Většina z nich využívá pro hledání cesty variace Dijkstrova algoritmu - A*star, který se jeví jako nejefektivnější. Z nich uvedu některé řešení na které jsem narazil.

3.1 PATHEngine

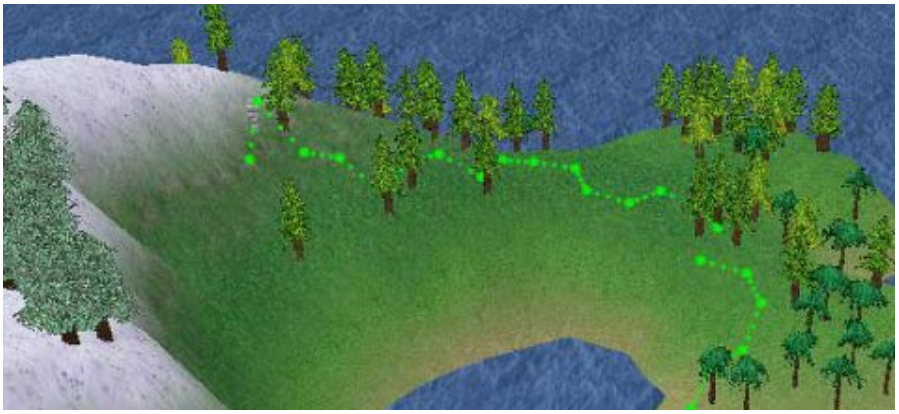
Je jednou z pokročilých knihoven pro hledání cesty pro komerčně vyvíjené hry. Knihovna je schopná plánovat trasy ve statickém terénu, ale je také schopna reagovat na dynamicky vytvářené překážky. Tyto překážky při hledání cest ihned zohledňuje v navigační síti cest. PATHEngine zároveň propojuje hledání cesty s kolizemi, aby dosáhl ideálních výsledných cest bez průchodů rohů překážek a podobných chyb. Knihovna by měla být schopna díky dobrému návrhu síťě cest hledat rychle cesty i na dlouhé vzdálenosti. Kvalitu knihovny dokazuje její využití v moderních komerčních hrách jako je hra vydána počátkem roku 2010 s názvem Metro 2033 od vývojové společnosti 4A Games. Viz reference [10].



Obrázek 1: Ukázka dema PATHEngine, mapa podle modelů.

3.2 MicroPather

MicroPather je opensource řešení pro hledání cesty postavená na algoritmu A*star implementovaná v jazyku C++. Je nezávislé na použité platformě a mělo by jí být možno snadno integrovat do kteréhokoliv stávajícího kódu hry. Autor se zaměřuje primárně na snadné použití, jednoduché API. Poskytuje úplné zdrojové kódy a hlavičkové soubory. Autor nezaměřuje svůj kód jako výukový ale pouze nabízí hotové řešení hledání cesty. Odkaz na manuál a zdrojové kódy najdete v referenci [11].



Obrázek 2: Ukázka dema MicroPather.

3.3 Autodesk Kynapse

Autodesk Kynapse je technologicky velmi pokročilou knihovnou poskytující API pro kompletní umělou inteligenci pro počítačové hry. Na této technologii je již dnes postaveno více než 100 herních titulů. Kynapse simuluje pokročilé plně dynamické hledání cest. Řeší problémy vnímání světa umělou inteligencí, pohyby velkých množství jednotek zároveň ve složitých terénech. Pomáhá také simulovat prostorové uvažování, týmovou spolupráci a obsahuje efektivní vývojové nástroje pro urychlení vývoje hry. Kynapse dokáže sestavovat mapy cest bez potřeby využít speciálního editoru cest. Tyto mapy cest se generují plně automaticky. Kynapse je poskytováno pro rozsáhlé portfolio platforem od herních konzolí Sony Playstation 3, Microsoft XBox 360,Nintendo Wii až po PC. Více informací viz reference [12].

Obrázek 3: Ukázka dema Autodesk Kynapse.

4 Specifikace zadání

Pro specifikaci požadavků počítačové hry je důležité nastínit představu o naší počítačové hře. Proto stručně shrnu naší představu počítačové hry v zadání a poté stanovím klíčové části, kterými se práce začne ubírat.

4.1 Zadání hry

Počítačová hra *Heroes*, na jejímž základu budeme v bakalářské práci pracovat, bude vycházet z kombinací klasického konceptu hry typu RPG, kdy hráč ovládá jediného hrdinu a strategické hry, ve které je záměr poražení nepřátelské strany zničením jeho hlavní budovy nebo důležité jednotky. Předpokládáme, že hráč bude podobně jako ve hře Diablo firmy Bilzzard, ovládat svého hrdinu myší a určovat jeho cíle cesty kliknutím do mapy. Postava na toto místo dojde nebo zaútočí na nepřátelskou postavu, která se na tomto místě nachází. Za zničené nepřátelské jednotky bude postava dostávat peníze, za které si bude dokupovat vylepšení. Herní mapa bude po technologické stránce podobná mapě strategické hry. Bude obsahovat množství překážek a její území bude rozděleno mezi jednotlivé strany, na kterých bude mít každá strana klíčovou budovu nebo jednotku, jejíž porážkou dojde k výhře jedné ze stran. Hra bude oživena tím, že jednou za určitý časový úsek vyjde z hlavních budov všech hráčů určité množství, na hráčích nezávislých, jednotek vyslaných na cizí základny. Cílem hrdinů (hráčů) bude pomoci těmto jednotkám v přesunu na nepřátelskou stranu. Zpravidla vítězí pak ti hráči, kterým se podaří probojovat až do nepřátelské základny, aniž by přišli o svou vlastní. Hra bude umožňovat, aby zároveň ve hře mohlo hrát více hráčů, kteří by měli každý svého hrdinu. Postavy vybíhající ze základen každé strany budou postupně silnější, aby se kompenzoval nárůst síly hrdinů. Tyto postavy se budou řídit jednoduchým plánem cest a předem budou mít nastavené své chování.

4.2 Specifikace požadavků

Na začátku softwarového procesu vývoje počítačové hry je třeba vytvořit specifikaci požadavků na části výsledné počítačové hry. Z našeho zadání hry vyplývá několik klíčových bodů, kterými se musíme na začátku vývoje hry zabývat, nalézt a ověřit správnost některých metod využívaných při vývoji počítačových her. V této bakalářské práci si beru za cíl nalezení řešení především těchto klíčových problémů :

- hledání cesty postav
- vyřešení síťové komunikace

Mezi další požadavky, které bude nutné vyřešit a které vyplývají ze zadání také patří řešení herního systému, který však bude složitější a jehož kompletní analýza a implementace bude součástí dalšího vývoje hry. Stejně jako vyřešení zvukového doprovodu hry.

4.2.1 Herní systém

Nad průběhem hry by měl dohlížet herní systém, který bude zajišťovat menu, vytváření hry, její ukončení a nastavování. Také by měl vyhodnocovat průběh hry a podle potřeby být schopný emitovat postavy umělé inteligence. Vyhodnocovat zásahy útoků a resetovat při smrti postavy hráčů do výchozí pozice. Podrobnější specifikaci jsem uvedl v kapitole 6 - Herní systém.

4.2.2 Umělá inteligence - Hledání cest

Ze zadání hry vyplývá že bude nutné vyřešit pro všechny postavy pohybující se ve hře problém hledání cest s přihlédnutím na rozmístění překážek. Postavy musí najít cestu mezi dvěmi body mapy a v ideálním případě by tato cesta měla být nejkratší a hledání by nemělo být časově náročné. Bude proto nutné vytvořit speciální mapu cest, na které budou zanesené všechny překážky, na které bude pak spuštěn jeden z algoritmů hledajících cestu. Podrobnější specifikaci najdete v kapitole 7 - Umělá inteligence .

4.2.3 Umělá inteligence - Inteligence postav

Ze zadání hry rovněž vyplývá že bude třeba vyřešit pohyb postav, které neřídí hráči. Bude třeba vymyslet jednoduchou umělou inteligenci AI, která bude realizována logickou sítí. Každá postava AI bude mít naplánovanou trasu a chování, kterého se bude držet. Později plánuji využití skriptovacího jazyka pro potřeby vytváření složitějších modelů chování.

4.2.4 Síťová komunikace

Podle zadání bude hra také umožňovat síťové hraní. U síťové komunikace se budu muset zaměřit na výběr síťového rozhraní, pak návrh architektury serveru a klientů. Pro komunikaci klient-server-klient vymyslet formát přenášených zpráv s informacemi o probíhající hře. Podrobnější specifikaci uvádím v kapitole 8 - Síťová komunikace.

5 Architektura enginu

Bakalářská práce je základem enginu pro počítačovou hru. Konkrétně se bude jednat o kombinaci Strategie/RPG a k tomu bylo nutné přizpůsobit jednotlivé části engine. Na každý druh počítačové hry jsou kladeny různé odlišné požadavky. U strategických her je kladen velký důraz na pohyb více jednotek najednou, a kvalitní umělou inteligenci. U her typu FPS, kde herní postava prochází uvnitř budovy se klade důraz hlavně na rychlost vykreslování a řeší se zde například specifický problém kolizí kulky s modelem. U arkádové plošinovky není zase nutné řešit hledání nejkratší cesty. Náš engine musí umět zajistit vykreslování 3D scény se statickými objekty a postavami řízenými hráči a postavy řízené umělou inteligencí. Pro všechny postavy je třeba hledat cesty mezi dvěmi body. A také je třeba zajistit chování postav neřízených člověkem. V případě síťového hraní je třeba zajistit přenos událostí mezi všemi připojenými klienty. Také je důležité předat hře informaci o akcích uživatele, aby mohl vůbec hru hrát. A v poslední řadě také zvukový doprovod. Proto jsme dospěli k následujícímu řešení rozdělení systému hry na moduly.

5.1 Specifikace požadavků

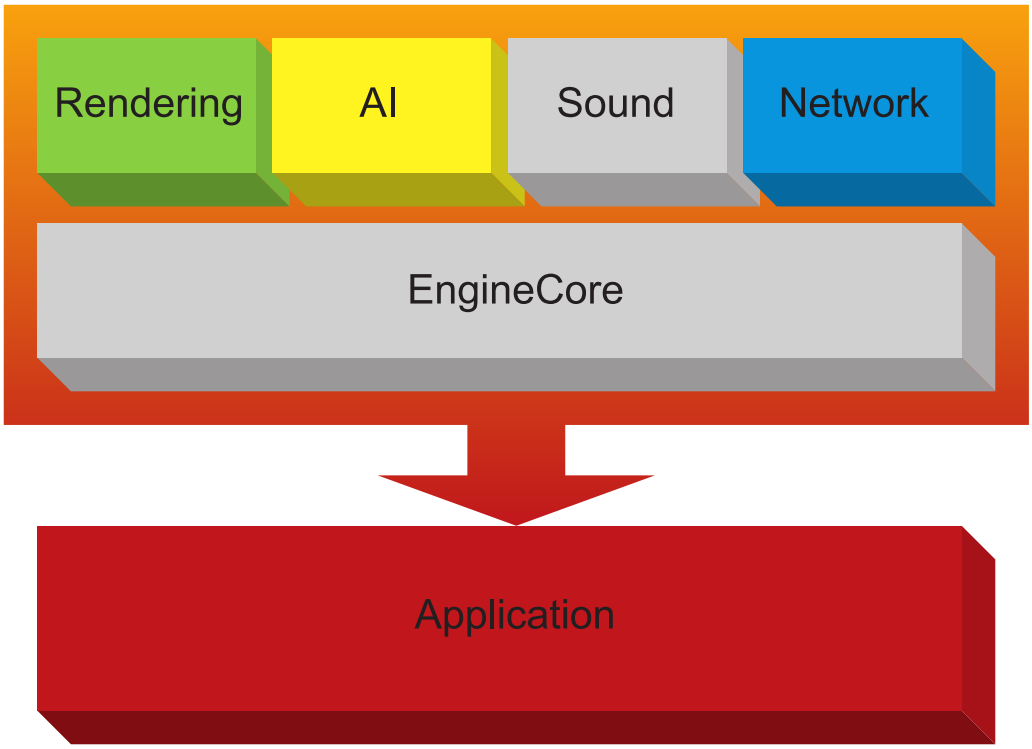
Budoucí engine bude muset spojovat jednotlivé části počítačové hry. Především jde o spojení částí vykreslování hry a herní logikou.

5.2 Návrh architektury

Pro engine počítačové hry jsme zvolili modulární schéma, kde každá konkrétní část hry má svojí vlastní dynamickou knihovnu, které mezi sebou komunikují pomocí herního enginu viz obrázek 4, který je poté vložen do samotné aplikace - hry. Z návrhu vyplynulo že bude třeba vytvořit následující moduly:

- modul Rendering - Mající na starosti vizuální stránku engine.
- modul AI - Zahrnující sestavení mapy cest, hledání cesty a řízení AI postav.
- modul Sound - Modul je knihovnu pro zpracování zvuku.
- modul Network - Zajišťuje síťovou komunikaci při hře více hráčů.
- propojující modul EngineCore - Zajišťující herní logiku a propojení jednotlivých modulů.

Moduly EngineCore a Sound nejsou prozatím plně implementovány, jejich implementace bude cílem práce až po dokončení dílčích částí, ale jsou v práci zmíněny.



Obrázek 4: Schéma architektury enginu.

6 Herní systém

Tato část bakalářské práce je dočasnou náhradou za neexistující část Engine Core a v budoucím vývoji z ní bude vytvořeno jádro celého enginu, který bude propojovat jednotlivé části. Pro představu funkce herního systému jsem specifikoval jeho požadavky. Herní systém jako takový musí zajišťovat celkový průběh hry viz diagram aktivit 6. Na začátku herní systém inicializuje instanci třídy, která se bude starat o probíhající hru. Zjistí zda si hráč přeje hrát síťovou hru, nebo ne. Na základě toho zajistí přes síťové rozhraní komunikaci s ostatními hráči. Za pomocí grafického modulu inicializuje grafiku a vytvoří virtuální svět ve kterém se hra bude odehrávat. Pro tento herní svět musí zajistit z modulu umělé inteligence spuštění generování mapy cest pro pohyb postav, zajistit vytvoření herní postavy každého hráče a vytvoření všech jednotek ovládaných umělou inteligencí. O všech těchto postavách musí informovat přes síťový modul všechny hráče. Poté musí herní systém spustit proces samotného hraní - zachytit události vytvořené hráčem a nebo činnosti umělé inteligence a rozesílat je všem připojeným hráčům. Dále se herní systém musí starat o sledování celého průběhu hry a v případě výhry jedné ze stran oznámit toto všem připojeným hráčům a hru ukončit.

6.1 Specifikace požadavků

Požadavky na herní systém jsem shrnul do těchto bodů:

- Po spuštění aplikace vytváří řídící třídu.
- Položí hráči dotaz na síťovou hru.
- V případě síťové hry připojí klienta pomocí rozhraní network k serveru.
- Zajistí, že první připojený hráč přebírá kontrolu nad hrou -startuje a ukončuje hru a zajišťuje vykonávání výpočtů a logiky umělé inteligence.
- V případě hry jednoho hráče probíhají tyto činnosti - spouštění hry, ukončení hry a činnosti umělé inteligence automaticky na lokálním stroji.
- Poté spustí grafickou knihovny a vytvoří herní svět.
- Pomocí AI knihovny vytvoří mapu cest pro navigaci na základě mapy světa s překážkami.
- Zajistí vytvoření seznamu postav hrdinů, postav umělé inteligence a v případě síťové hry tyto informace ihned přeposílá ostatním klientům.
- Vyšle do sítě připojených klientů povel ke startu samotné hry.
- V každém cyklu vykreslování grafiky zachytí události hráče a agentů umělé inteligence a pokud je hra síťová, pak tyto události přeposílá, přijímá a upravuje parametry jednotek v průběhu hry.

- Pro celou hru provádí kontrolu stavu jednotek, vyhodnocuje případnou prohru nebo výhru jedné ze stran. Oznamuje výsledky síťovým hráčům a ukončuje jejich hru.

6.1.1 Správa hry

Pro správu průběhu hry bude v budoucí herní implementaci jádra nutné udržovat informace o aktuálním stavu všech proměnných na jednom místě. Pro implementaci počítám s řídící třídou, která bude obsahovat sadu atributů na jejichž základě se bude hra řídit. Schéma průběhu hry najdeme na diagramu aktivit viz obrázek 6. Třída bude implementovat také kontrolní funkce, které budou spouštět hru a ukončí hru při splnění podmínky výhry. Třída zároveň bude zahrnovat seznam jednotek na mapě, informaci o tom zda je připojený uživatel označen jako "master", což bude znamenat že na jeho počítači bude probíhat vyhodnocování umělé inteligence a také bude spouštět hru. Mezi dalšími důležitými parametry budou: IP adresa serveru, jméno hráče, identifikační číslo hráče přidělené serverem, informace o straně za kterou hráč hraje, informaci zda je hra síťová a seznam všech jednotek vyskytujících se na herní mapě. Pro přibližné nastínění řídící třídy jsem vypracoval třídní diagram viz obrázek 5 shrnující plánovanou implementaci.

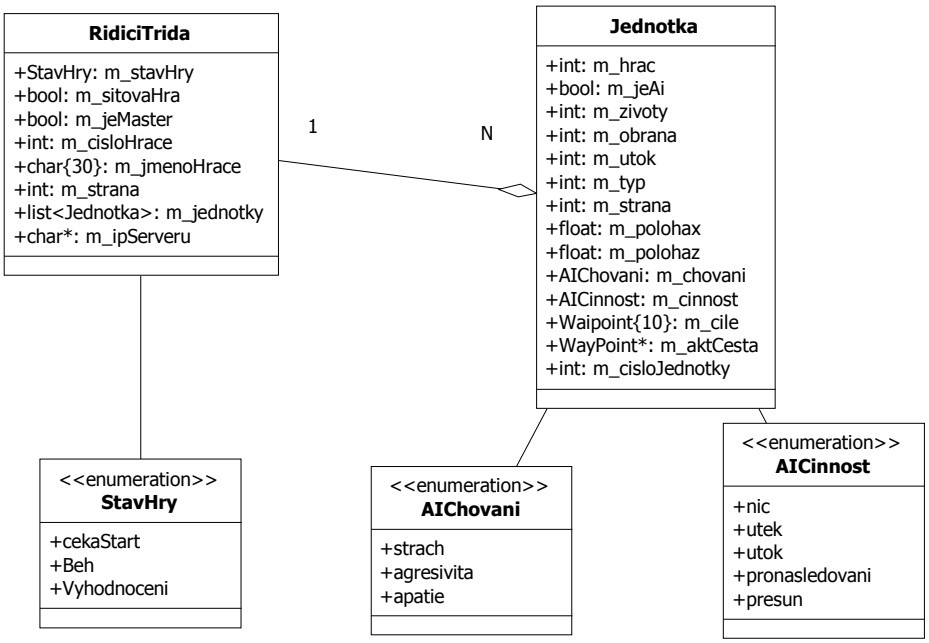
6.1.2 Správa jednotek

Herní systém bude muset udržovat seznam jednotek, které se vyskytují ve hře. Seznam bude obsahovat nejen jednotky umělé inteligence, ale i postavy hráčů a nebude problém jej do budoucna rozšířit i pro budovy. Každá jednotka bude uložena v seznamu řídící třídy. Výčet parametrů jednotky je uveden v třídním diagramu viz obrázek 5, který shrnuje klíčové atributy každé jednotky. Pro přehled uvedu jejich výčet.

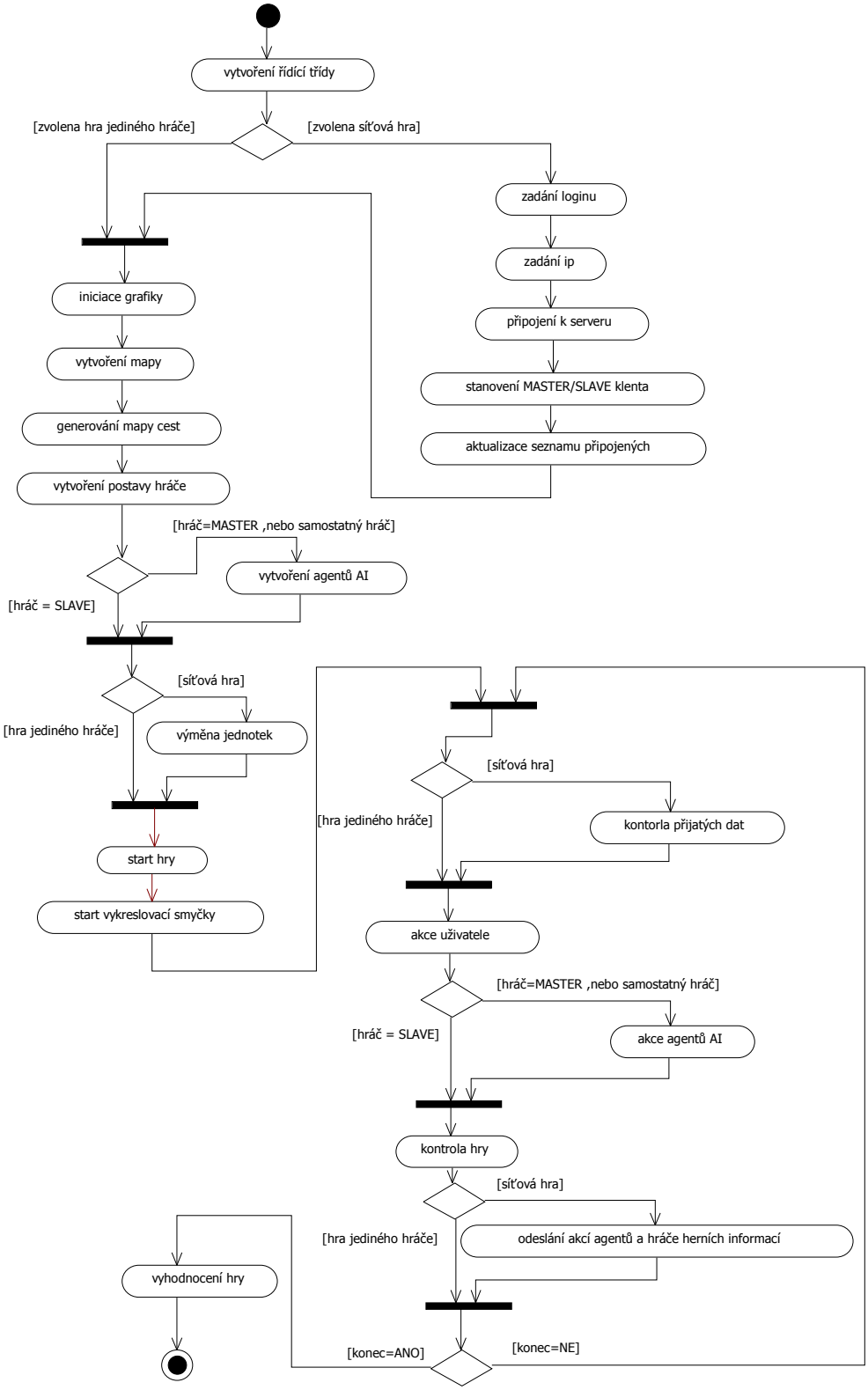
- Číslo hráče a číslo jednotky.
- Informaci, zda je postava řízena umělou inteligencí.
- Obranu, útok, počet životů, polohu postavy na mapě.
- Typ postavy. V pozdější plánované implementaci hry bude výčet možných postav, podle něj se bude přiřazovat správný 3D model.
- Informaci o straně za kterou postava hraje.
- Pro potřeby umělé inteligence bude jednotka nést informace o svém chování a činnosti.
- Pro plánování trasy umělé inteligenci bude nést jednotka seznam deseti cílových bodů cesty, podle nichž se bude řídit.
- Pro pohyb bude jednotka nést seznam bodů cesty.

6.2 Shrnutí

Řízení průběhu hry je neméně důležitou a často opomíjenou nutností v počítačové hře. Díky návrhu herního systému jsem si uvědomil jaké problémy se musí řešit v samotném jádru hry. Stručný popis činnosti systému a návrhu struktury programu jsem popsal v diagramech 6 a 5.



Obrázek 5: Třídní diagram řídící třídy a třídy Jednotka.



Obrázek 6: Diagram aktivit pro herní systém.

7 Umělá inteligence

Tato část bakalářské práce se zabývá návrhem bloku AI, který bude zajišťovat všechny činnosti spjaté s umělou inteligencí. Největším problémem umělé inteligence ve strategických hrách je navigace a herní logika.

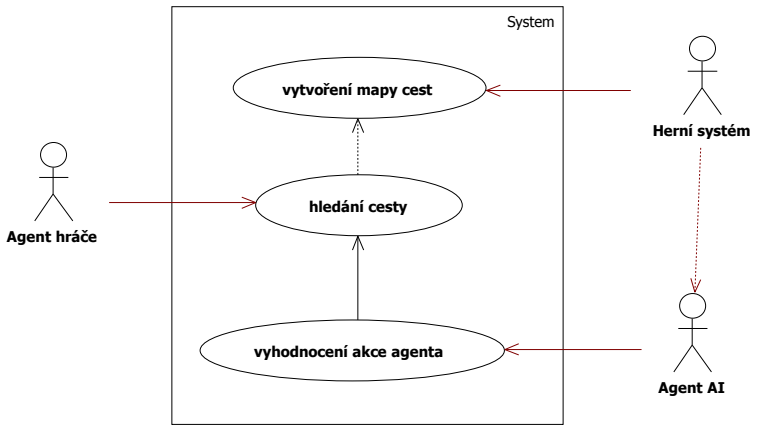
7.1 Umělá inteligence - hledání cesty

7.1.1 Specifikace požadavků

Blok umělé inteligence bude dle specifikace zadání zajišťovat následující funkcionalitu:

1. Sestavení mapy cest na základě existující herní mapy s překážkami.
2. Na mapě cest realizuje hledání nejkratší cesty mezi dvěma body.
3. Zajistí hledání většího množství cest v jeden okamžik nezávisle na sobě.
4. Vyhodnocuje stav počítačem řízených agentů ve hře a následné vyvolává akcí na základě jednoduché logické sítě (inteligentním jednání).

Pro specifikaci chování agentů bylo vhodné sestavit diagram případů užití obrázek 7, který k výčtu funkcí doplňuje i jejich aktéry.

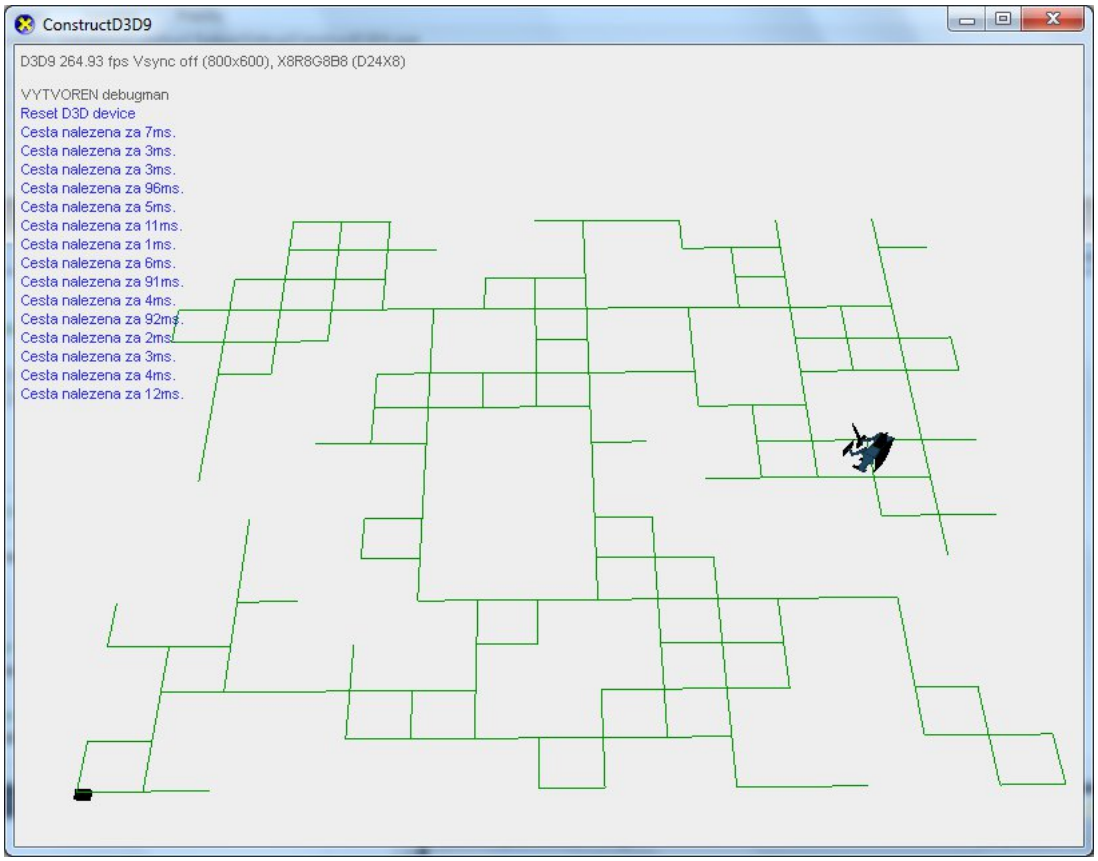


Obrázek 7: Diagram případu užití pro blok umělé inteligence.

7.1.2 Analýza a návrh implementace

V následujících kapitolách je popsána analýza a návrh implementace bloku umělé inteligence pro hledání cesty.

7.1.2.1 Sestavení mapy cest Mapa cest je pro navigaci herních postav ve hře velmi důležitou součástí herního světa. Není nutné aby tato struktura byla zobrazována, avšak algoritmy hledání cesty jí potřebují. V mém návrhu budu počítat s jednoduchou čtvercovou sítí, ale později plánuji realizaci generátoru, který vytvoří optimalizovanou síť cest s menší hustotou, což by mělo vliv na značné urychlení celého algoritmu hledání. Můj návrh počítá s ohodnocením každé hrany mezi dvěmi body mapy cest hodnotou jedna. Ve čtvercové síti pak pro každý uzel platí, že má 0 až 4 sousedy. Ukázka takto vytvořené struktury je na obrázku 8, na kterém prázdné mezery ve mřížce značí místa kde je překážka. Strukturu mapy cest popisuje diagram tříd viz 10.



Obrázek 8: Ukázka mapy cest.

7.1.2.2 Hledání cesty Hledání cesty je jednou z nejpodstatnějších částí umělé inteligence. Hráč výběrem cílového místa určí kam postava má jít a ta se na toto místo začne přesunovat. Existují dvě možnosti řešení tohoto problému. Jedním z nich je nehledat nejkratší cestu, ale zjišťovat polohu překážky před postavou a podle určitých pravidel zvolit směr obcházení. Toto řešení je však ve složitějších mapách špatné - postavy nelogicky jdou do slepé ulice a poté se vrací, nebo dokonce zůstanou ve slepé ulici stát

a dál se nepohybují. Také může zvolený směr obcházení překážky být nelogický a postava pak musí ujít několikanásobně delší vzdálenost. Tyto způsoby řešení se jeví na pohled hráče hry nelogické. Druhým způsobem je použití mapy cest. Mapa cest definuje na herní mapě cesty, kterými se postava může vydat. Po určení cílového místa se spustí algoritmus hledání cesty, který projde body v mapě cest od aktuální pozice po cílovou pozici a vyhledá nejkratší cestu i s ohledem na hustou síť překážek. Postava cestující po této cestě již vypadá na pohled inteligentně. V této práci se zabývám řešením hledání cest ve statické mapě, ale později hodlám zohlednit i pohybující se objekty, aby nedocházelo k průchodům míjejících se postav přes sebe. V podstatě se jedná o kombinaci prvního způsobu obcházení překážky podle určitého pravidla pro dynamické objekty ve hře (míjející se, nebo souběžně jdoucí postavy) a zároveň se při pohybu držet cesty vygenerované algoritmem.

7.1.2.3 Algoritmy hledání cesty Pro hledání cesty v grafech s nezápornou délkou hran se obecně používá Dijkstrův algoritmus viz [2]. Při jeho průchodu grafem algoritmus prohledává uzly do šířky od výchozího bodu, dokud nenajde cílový uzel. Nejvyšší časový odhad algoritmu je dle knihy [2] $O(n^2+m)$. Prohledávání však zahrnuje procházení uzlů ve všech směrech, a to zvyšuje časovou náročnost celého hledání. Pro hledání cest v naší počítačové hře je konkrétně samotný algoritmus Dijkstra nepoužitelný. Pro nalezení optimální cesty bylo proto nutno využít jiný algoritmus, který by při hledání cesty zohledňoval i směr procházení.

7.1.2.4 Výběr algoritmu A*star Pro hledání cesty grafech jsem se rozhodl využít Dijkstrův algoritmus doplněný o heuristický výpočet, jelikož se jedná o nejlepší známý algoritmus řešící tento problém. Vysvětlením Dijkstrova algoritmu i Dijkstrova algoritmu s heuristikou se zabývá kniha [2] v kapitole 6.4,6.5 a 6.6. V anglické literatuře je algoritmus známý pod názvem A*star. Díky heuristické funkci algoritmus nemusí prohledávat celý graf, a pořadí prohledávání uzlů je upravováno odhadem vzdálenosti k cílovému bodu. Toto chování je oproti algoritmu Dijkstra vidět z obrázku převzatého z knihy [3], Dijkstrův algoritmus prochází celou plochu - obrázek vpravo. Probíhá prohledávání do šířky. Algoritmus A*star prochází díky heuristické funkci založené na absolutní vzdálenosti od cíle menší počet uzlů - v obrázku vlevo. Vychází z toho že známe směr, kterým leží cíl.

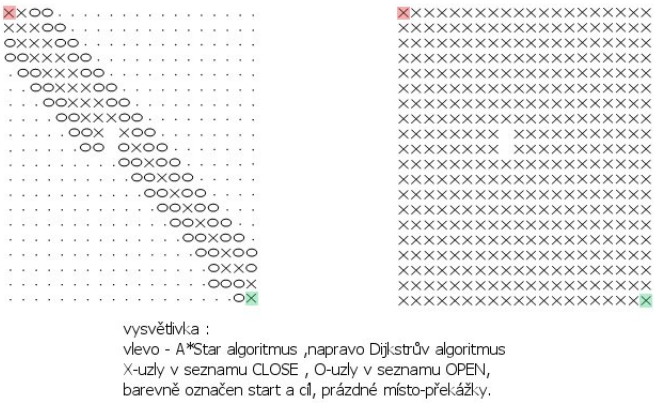
Hodnotící funkci uzlu x popisuje výpočet :

- f(x) je předpokládaná délka cesty
- g(x) je uražená vzdálenost z výchozího bodu
- h(x) je hodnota odhadované vzdálenosti k cíli (heuristika)

předpokládanou délky cesty vypočteme jako f(x)=g(x)+h(x).

7.1.2.5 Popis algoritmu A*star Pro sestavování cesty pomocí algoritmu A*star uvedu stručný popis jeho chování:

1. Vytvoř seznamy OPEN, CLOSE a pro zapamatování prošlých uzlů a ukazatel aktuálního uzlu U.
2. Vypočti hodnoty f(x), g(x) a h(x) počátečního uzlu START.
3. Vlož uzel START do seznamu OPEN.
4. Vyhledej v seznamu OPEN uzel s nejnižší hodnotou f(x), označ jej jako U a vlož jej do seznamu CLOSE.
5. Pokud je aktuální uzel U = CÍL ukonči hledání.
6. Pro všechny sousední uzly okolo U, které nejsou v seznamech OPEN, ani CLOSE vypočti hodnoty f(x), g(x) a h(x) a přesuň je do seznamu OPEN.
7. Dokud není seznam OPEN prázdný opakuj bod č.4 jinak pokračuj k dalšímu bodu
8. seznam CLOSE nyní obsahuje cestu k od cíle ke startu
9. procházej seznam close od konce a vybírej vždy uzel s hodnotou g(x)-1 a kopíruj je do nového seznamu lifo - Path(cesta).



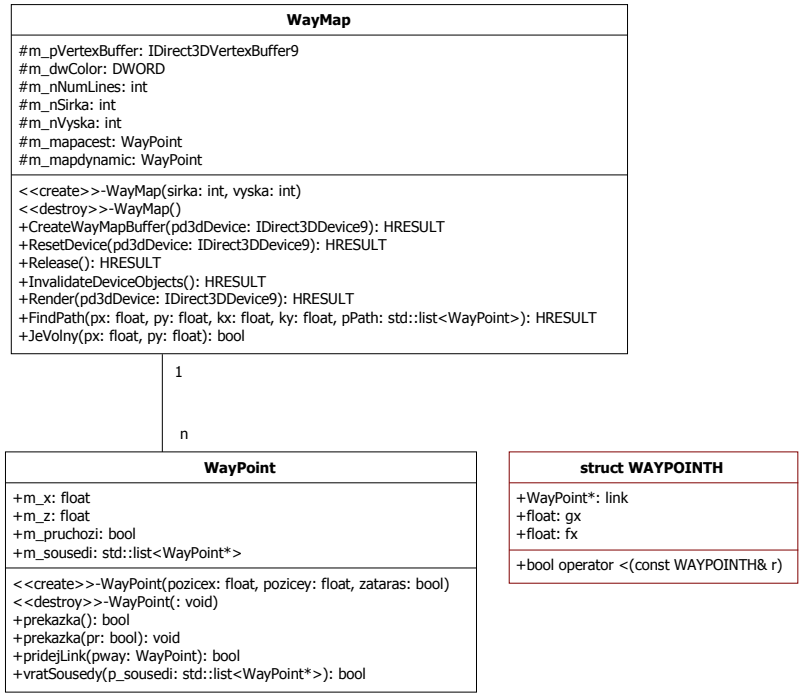
Obrázek 9: Ukázka rozdílu mezi algoritmem Dijkstra a A*star

7.1.2.6 Složitost algoritmu A*star Časová složitost algoritmu závisí na heuristické funkci. V nejhorším případě je složitost $O(n^2 + m)$, kdy je stejná jako u Dijkstrova algoritmu. K tomu dojde tehdy, není li cílový uzel cesty v grafu a musí se proto prohledávat celá mapa cest do šířky. Viz [2].

7.1.2.7 Návrh implementace hledání Návrh implementace algoritmu A*star vychází ze stanovených vlastností herního světa, u kterého jsem si určil tyto důležité vlastnosti :

- Mapa cest je určena čtvercovou mřížkou.
- Mřížka obsahuje uzly, které mají atribut průchodnosti.
- Vzdálenost uzlů je rovna hodnotě 1 v herním světě.
- Z každého uzlu jsou odkazem připojeny další 4 uzly, kromě okrajů mapy cest.
- Každý uzel má udány své souřadnice ve 2D prostoru plochy mapy.

Implementace hledání je provedena na struktuře tříd, kterou udává následující třídní diagram 10. Mapa cest WayMap je implementována jako pole instancí WayPoint, které jsou na sebe vzájemně navázány pomocí odkazů.



Obrázek 10: Třídní diagram pro hledání cesty

7.1.2.8 Urychlení běhu hledání cest Následující úpravy jsem provedl na struktuře mapy a také na samotném algoritmu A*star abych docílil zrychlení jeho běhu.

Mapa WayMap je vytvořena polem instancí třídy WayPoint. Každá instance si nese v seznamu std::list adresy instancí svých sousedů. Díky tomu při prohledávání mapy

neadresují přímo jednotlivé pole WayMap souřadnicemi na mapě, ale mezi jednotlivými instancemi se pohybuji pomocí odkazů, které si u sebe instance Waypoint nese. Díky tomu je v podstatě vyhledávací algoritmus nezávislý na struktuře mapy a ta by teoreticky mohla být tvořena trojúhelníky nebo hvězdami i jinými propojenými tvary. Na každý takový WayPoint v mapě může být proto navázán i větší počet cest k libovolným Waypointům na celé mapě. Toto řešení mi přišlo vhodné z důvodu pozdějšího vylepšování vyhledávacího algoritmu a opuštění klasické čtvercové mřížky. I když přiznávám, že je možné že díky tomuto konkrétnímu řešení jsem prohledávání grafu zpomalil. Prohledávání souřadnicemi v poli totiž umožňuje přímé adresování kterékoliv buňky na mapě. Jedno z pozitiv mého řešení je to, že se nemusím při adresování zabývat problémem možného adresování buňky mimo mapu, kdy v C++ obvykle toto adresování způsobuje vyjímku a nečekaný pád programu. Navíc se tyto chyby obvykle špatně odhalují. Mapa je zároveň dvojrozměrné pole a proto je možné i přímé adresování jednotlivých uzlů. Tato vlastnost se dá využít při zjišťování u kterého uzlu zrovna postava stojí, pokud známe její souřadnice.

Urychlení činnosti algoritmu A*star jsem dosáhl pomocí úpravy seznamu OPEN, dle kapitoly 7.1.2.5, který nese seznam všech otevřených bodů. Oba seznamy OPEN i CLOSE nesou v mé implementaci instanci struktury WAYPOINTH, která obsahuje odkaz na konkrétní WayPoint a každý uzel mapy ještě hodnotící funkce g(x) a f(x). K optimalizaci dojde tím, že při vkládání instancí WAYPOINTH do seznamu OPEN jsou tyto instance ihned tříděny podle hodnoty f(x). Této vlastnosti jsem pro seznam OPEN dosáhl využitím tříděného seznamu std::set. Seznam má implementováno při přidávání instancí třídění quick short a pro správnou funkci na mé instanci WAYPOINTH stačilo jen dodatečně přetížení operátoru <. Algoritmus poté nemusí sekvenčně prohledávat celý seznam OPEN, ale přímo vybere WayPoint s nejnižší hodnotou f(x), okolo kterého probíhá dle kapitoly 7.1.2.5 vyhledávání dalších bodů.

Zpětné sestavení cesty už jen zkopíruje odkazy na uzly ze seznamu CLOSE do nového pole list Path nyní už typu WayPoint. Kopíruje se však od cílového uzlu směrem ke startu a při kopírování se vybírají ty body, které mají hodnotu g(x) vždy o jedna menší. U každého z nich se kontroluje seznam sousedů, jestli je opravdu sousedním bodem. Takto se sestaví cesta až ke startu a může se předat postavě.

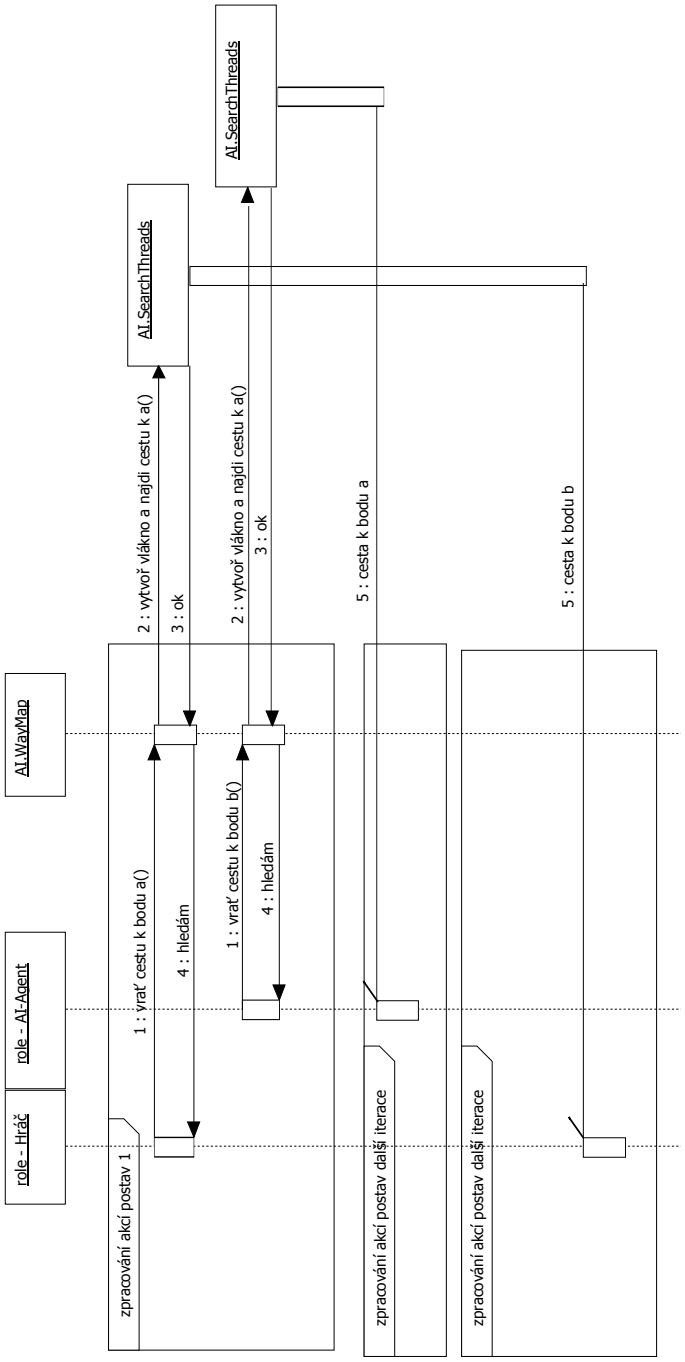
7.1.2.9 Souběžné hledání cest V počítačové hře bude potřeba současně vyhledávat cesty pro více postav najednou. Vyhledávání cesty je časově náročná operace a není vhodné jí zdržovat hlavní vykreslovací vlákno programu. Proto jsem se rozhodl při hledání cesty využít možnosti hledat cestu nezávisle na hlavním vláknu, a to pomocí vláken, konkrétně pomocí knihovny ZThread. Je totiž možné že bude potřeba v jednom okamžiků více hledání cest současně, což by mělo za následek při překročení času 40ms (při 25 FPS) znatelné záseky vykreslovací smyčky. Vytvořením vlákna sice dojde k drobné časové ztrátě, avšak výsledek hledání cesty není důležité dostat ihned a hráč by ani neměl rozeznat prodlevu mezi stiskem tlačítka myši a začátkem přesunování postavy na požadované místo i při několikanásobném překročení hledacího času do 40ms. Také tato doba bude

silně závislá na požadovaném cíli cesty. Pokud tedy hráč půjde blízko bude vyhledávací doba také velmi nízká a hráč prodlevu téměř nepostřehne. Vzdálenost kam postava hráče může jít bude navíc limitována pohledovým objemem kamery. To znamená, že hráč může jít jen tam kam dohlédne. Extrémní případ je, že hráč bude požadovat chůzi do překážky. V tento moment musí hledání cesty projít všechny body herní mapy aby zjistilo že cíl je nedostupný a v závislosti na velikosti mapy bude také toto hledání patřičně časově náročné. I pro tento případ je vytvoření vedlejšího vlákna výhodné. Souběžné hledání bude třeba hlavně kvůli agentům umělé inteligence, kteří mohou cestovat nezávisle na velikosti pohledového objemu kamkoliv na mapě. O vyvolání tohoto hledání se bude starat logická síť - kapitola číslo 7.2.

Postup vytváření vláken pro hledání zachycuje sekvenčním diagram 11. Kterákoliv postava ve hře může požadovat vyhledání cesty. Díky tomu, že Zthread implementuje zásobník vláken je možné ušetřit čas vyvolávání vlákna. Vlákna jsou alokována na začátku programu a poté jsou díky ním zpracovány úkoly dílčích hledání cest. Po dokončení úkolu se vlákno opět přidá do zásobníku.

7.1.3 Testování

Testování implementace algoritmu A*star jsem se rozhodl provést na mapě cest o rozměrech 128 x 128 bodů, celkem na 16384 uzlech a 64012 hranách. V budoucím herním světě jde o průměrně velkou mapu. Výsledky rychlosti hledání pro jedno samostatné vlákno jsou dány výkonem procesoru testovaného počítače, viz další kapitola. V mém testu jsem testoval vliv množství překážek v terénu na rychlost vyhledávání. Pro test jsem si vybral konkrétně 0%, 10% a 20% náhodně rozmístěných překážek ku průchozím uzlům. Ve výsledku překážka znamená ztrátu cest - hran okolo uzlu, který není průchozí. V grafu nejsou zanesené hodnoty pro hledání, v případě že cílový uzel není z výchozího uzlu dostupný. Pro 0% překážek tato hodnota neexistuje, protože všechny uzly jsou dostupné. Pro 10% jde o čas přibližně 5400 ms. Pro 20% jde o čas přibližně 3200ms. Čas pro větší procento překážek je nižší, protože ubyl počet uzlů na procházení i když opticky vypadá struktura složitěji. Celkové výsledky jsem uvedl v grafu 18, v přílohách. Podle výsledků sady náhodných měření jsem do grafu vynesl křivky délky cesty a průměrných hodnot potřebného výpočetního času. Graf udává jen hodnoty naměřené pokud existovala cesta z počátečního uzlu do koncového. V herním světě bez překážek je algoritmus nejrychlejší. Při 10% překážek byly časy nejhorší a na velké vzdálenosti překročily 100ms. Při překročení času 40ms přitom pozorovatel již vidí krátké zastavení pohybu postavy. Pro 20% zastoupení překážek ku průchozím uzlům se běh opět začíná zrychlovat s ubývajícím počtem uzlů k testování. S výsledky měření jsem spokojen, i když jsem očekával lepší časy. Bude proto potřeba vymyslet ještě další optimalizace. Jako jedna z nich je možnost upravení heuristické funkce. Další možností je vypouštění nedostupných bodů. To zajistím tak, že při potřebě hledání cílového nedostupného uzlu první spustím hledání algoritmem Dijkstra za účelem zjištění nejbližšího dosažitelného uzlu. Poté již hledám cestu k existujícímu cíli. Což by mělo za následek eliminaci prohledávání celé



Obrázek 11: Sekvenční diagram vytváření vláken.

mapy cest. A značný nárůst výkonu. Výhoda implementovaného řešení je při cestování na krátké vzdálenosti. Je to minimální hledací doba.

7.1.3.1 Testovací sestava Testování probíhalo na mém osobním počítači o těchto parametrech. Uvádím jen informace, které mají vliv na výkon testované aplikace.

- Dvoujádrový procesor AMD Athlon X2 3600+ 1.9GHz
- 2GB operační paměti DDR2, 800MHz.
- Operační systém Microsoft Windows 7 Profesional 64bit.
- Grafická karta AMD-ATI radeon 2600XT 256MB.

7.2 Umělá inteligence - řízení agentů

Problém inteligentního řízení postav není hlavním cílem této bakalářské práce a mohl by být tématem na samostatnou bakalářskou práci. Umělou inteligencí se zabývá celý obor informatiky, který se snaží vytvořit model "inteligentního chování". Jako etalon inteligentního chování bývá uváděn lidský rozum. Existuje mnoho způsobů jak simulovat chování postav umělé inteligence. Velká většina komerčních her staví svou umělou inteligenci na kombinaci dynamického hledání cest a logické síti implementované v některém z mnoha skriptovacích jazyků. Tyto skriptovací jazyky umožňují postavě nadefinovat chování při situacích, které se ve hře mohou nastat. Herní engine předává takovému skriptu polohu a informaci o tom, co postava vidí a skript definuje co postava ve hře má dělat, jaký je její cíl a její chování na základě těchto vstupů. Takovéto inteligence se implementují dnes do většiny počítačových her a bude součástí i naší budoucí implementace. Avšak nejedná se o inteligenci v pravém slova smyslu. Takováto inteligence nedokáže reagovat na novou situaci, která není v naskriptována. Většina inteligencí pro počítačové hry nejsou nic jiného než logické sítě bez možnosti se učit nové informace a pouze se drží naimplementovaného chování. Oproti tomu existují i počítačové hry které šly v umělé inteligenci dále a implementovali pro chování postav neuronovou síť. Základním principem neuronových sítí jsou jejich stavební prvky neurony, které fungují jako logické obvody řízené váhami. Suma všech podnětů na receptory neuronů vyvolá dle nastavené váhy reakci. Na receptory se přivádí kromě podnětů z herního světa i zpětné vazby, které umožňují realizovat paměť. Nejdůležitější je však v této síti možnost neuronovou síť dynamicky upravovat, rozšiřovat o nové neurony a vazby. Síť je poté schopna učení. Příkladem takovéto hry je počítačová hra Black and White z roku 2001, ve které ovládáte postavu hráčova pomocníka, která je řízena neuronovou sítí a je schopna se naučit některým činnostem. Například když herní postava sní kámen, zjistí že kámen jí nezažene hlad a vyplivne ho. Příště už se o pojídání kamene nebude pokoušet. Pamatuje si své chybné jednání a na základě jej se učí.

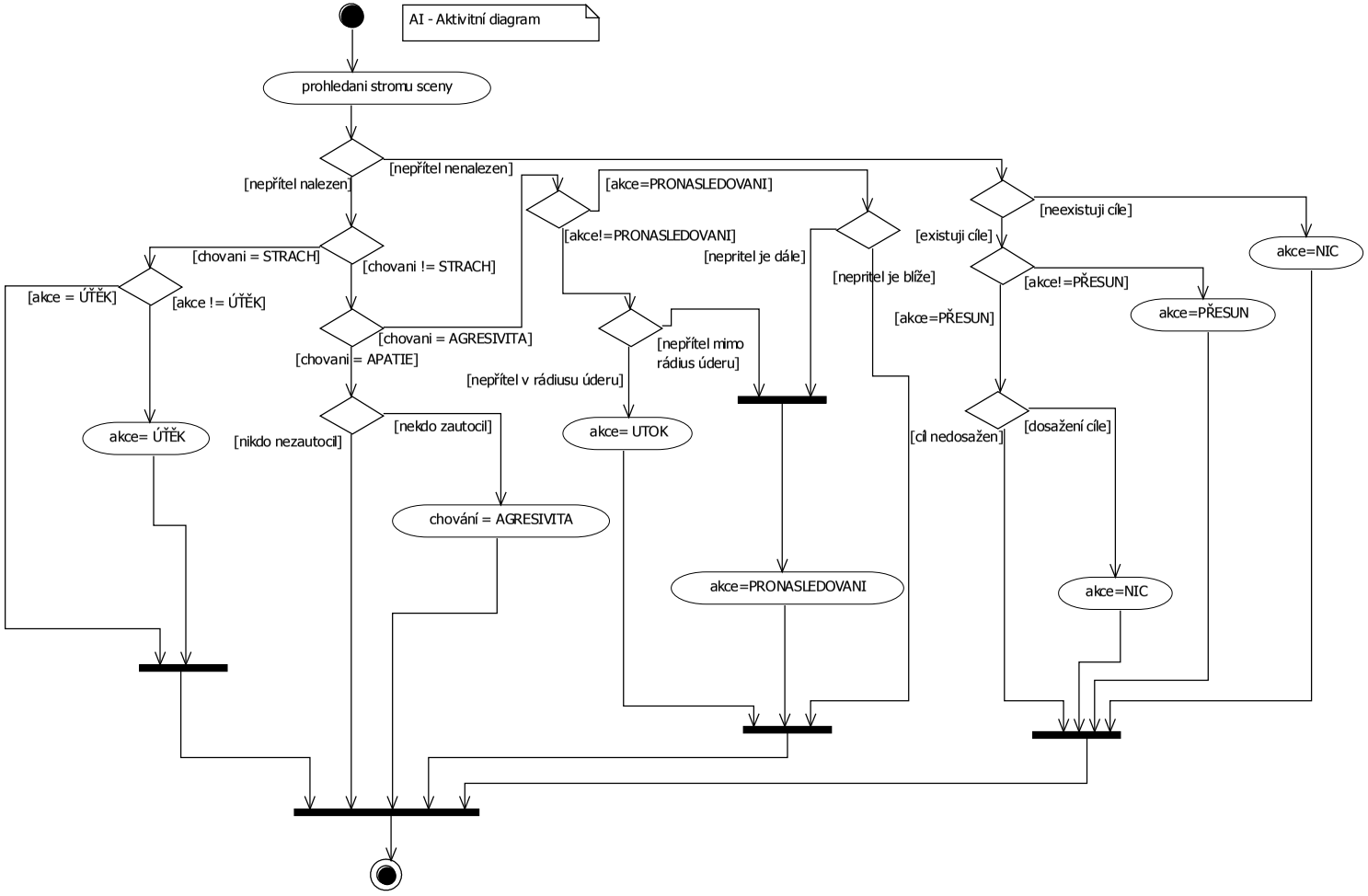
7.2.1 Specifikace požadavků

V budoucí počítačové hře bude umělá inteligence řízena jednoduchou logickou sítí. Implementace bude realizována až v pozdější fázi vývoje počítačové hry neboť je problém vývoje hry velmi komplexní. Má specifikace umělé inteligence počítá s těmito základními požadavky na postavy neřízené člověkem:

- Vytváření postav agentů a stanovení jejich cílů a chování ve hře.
- Postava nezávisle na svém chování vykonává naplánované cíle, cíle uvažuji jako body trasy, které má postava projít.
- Každý agent může mít nastaveno chování stupni strach, agresivita a apatie
- Strach - Postava s modelem chování strach se snaží utéct od nebezpečí nepřátelské jednotky, po poklesu životů pod určitou hladinu se mění na postavu agresivní.
- Agresivita - Postava s tímto modelem chování napadá veškeré nepřátelské jednotky nebo budovy které potká.
- Apatie - Postava s tímto modelem chování se striktně drží stanovených cílů a neútočí na nepřátelské jednotky, pokud je na postavu veden útok okamžitě se mění její chování na agresivitu.
- Pro každou postavu je kromě modelu chování stanoven i seznam akcí, které může vykonávat nezávisle na cílech. Jsou to akce :nic, útěk, útok, pronásledování, přesun. Mezi těmito akcemi postava neustále střídá na základě výskytu nepřátelských jednotek a modelu chování.
- Nic - Postava nevykonává žádnou akci, stojí na místě a vyčkává.
- Útěk - Postava s chováním strach se snaží utéct co nejdál od nepřítele.
- Útok - Postava s chováním agresivita útočí na nalezeného nepřítele.
- Pronásledování - Postava s modelem chování agresivita se snaží pronásledovat nejbližšího nepřítele.
- Přesun - Postavy s kterýmkoliv modelem chování se přesouvají na místo stanovené cíli.
- Při střetu postavy s nepřátelskou jednotkou se provede výběr akcí na základě stanoveného chování a pak dále pokračuje v plnění stanovených cílů.

Výčet všech možností je možné dále rozšiřovat a přidávat nové modely chování a činnosti podle nich vyvolanými. V podstatě je možné že bude v budoucnosti nutné model chování přepracovat a doplnit o složitější akce a celý systém implementovat skriptovacím jazykem. Díky mému řešení by však postavy umělé inteligence měly být později schopny

například vycházet z hlavní budovy nepřítele a být ve vlnách poslány na přátelské budovy, nebo hlídkování na předem stanovených místech. Pro tyto účely se řešení jeví jako dostačující.Popisovanou strukturu logické sítě shrnuje následující diagram aktivit 12 na kterém jsem shrnul přechody mezi jednotlivými chováními a akcemi postav umělé inteligence.



Obrázek 12: Diagram aktivit pro agenty.

8 Síťová komunikace

V dnešních počítačových hrách je možnost hrát hry společně s přáteli již téměř nutností. Už od počátku počítačové techniky byly vždy hry, které si můžete zahrát proti, nebo s kamarádem o mnoho zábavnější. Hráči si sami pomáhají dotvářet herní svět svou vlastní interakcí a fantazií. Veškeré hraní se postupně přesunuje na celosvětovou počítačovou síť Internet.

Tato část bakalářské práce se věnuje výměně informací v probíhající počítačové hře přes sítě LAN a Internet. Stanovuje požadavky na síťový modul, vybírá technologie na kterých je síťování hry postaveno a ukazuje mou představu o správné síťové topologii do her.

8.1 Specifikace požadavků

V našem technologickém demu bylo potřeba vyřešit několik základních otázek týkajících se síťové komunikace jednotlivých klientů. Mezi hlavní otázky patří následující výčet.

- Jakou zvolit topologii?
- Jakou zvolit síťovou technologii?
- Jakou komunikaci navázat mezi serverem a klientem ?
- Zvolit chytrý server, nebo jednodušší?
- Jak přenášet data mezi klienty - formát zpráv?

Na všechny otázky bylo třeba najít odpověď a stanovit nejlepší řešení.

8.2 Analýza a návrh implementace

Následující kapitoly rozebírají analýzu a návrh implementace jednotlivých částí síťové komunikace.

8.2.1 Topologie sítě

Při volbě topologie jsem zohlednil své předchozí nabyté znalosti z předmětu počítačových sítí a rozhodl se pro implementaci topologie hvězdy na aplikační vrstvě referenčního modelu ISO/OSI. Topologie je tvořena jedním serverem,který přeposílá data mezi klienty. Toto řešení je výhodné zejména kvůli menšímu počtu spojení a také kvůli tomu že server může běžet na počítači s veřejnou ip adresou a může se k němu připojit v podstatě kdokoliv z internetu, nebo i v rámci lokálních sítí.

8.2.2 Síťová technologie

V zadání jsme měli uvedeno zvážení technologie DirectPlay, která je součásti knihoven DirectX, ale v oficiální dokumentaci MSDN na stránkách Microsoftu viz. reference [5] jsem se dočetl že se v nových verzích DirectX od používání DirectPlay upouští a doporučuje se implementovat komunikaci ve standardních knihovnách Winsock2 (verze2.2) obsažených ve Windows. Proto jsem se rozhodl implementovat komunikaci tímto způsobem. Winsock2 uvádím v kapitole použitých technologií.

8.2.3 Volba mezi TCP-UDP komunikací

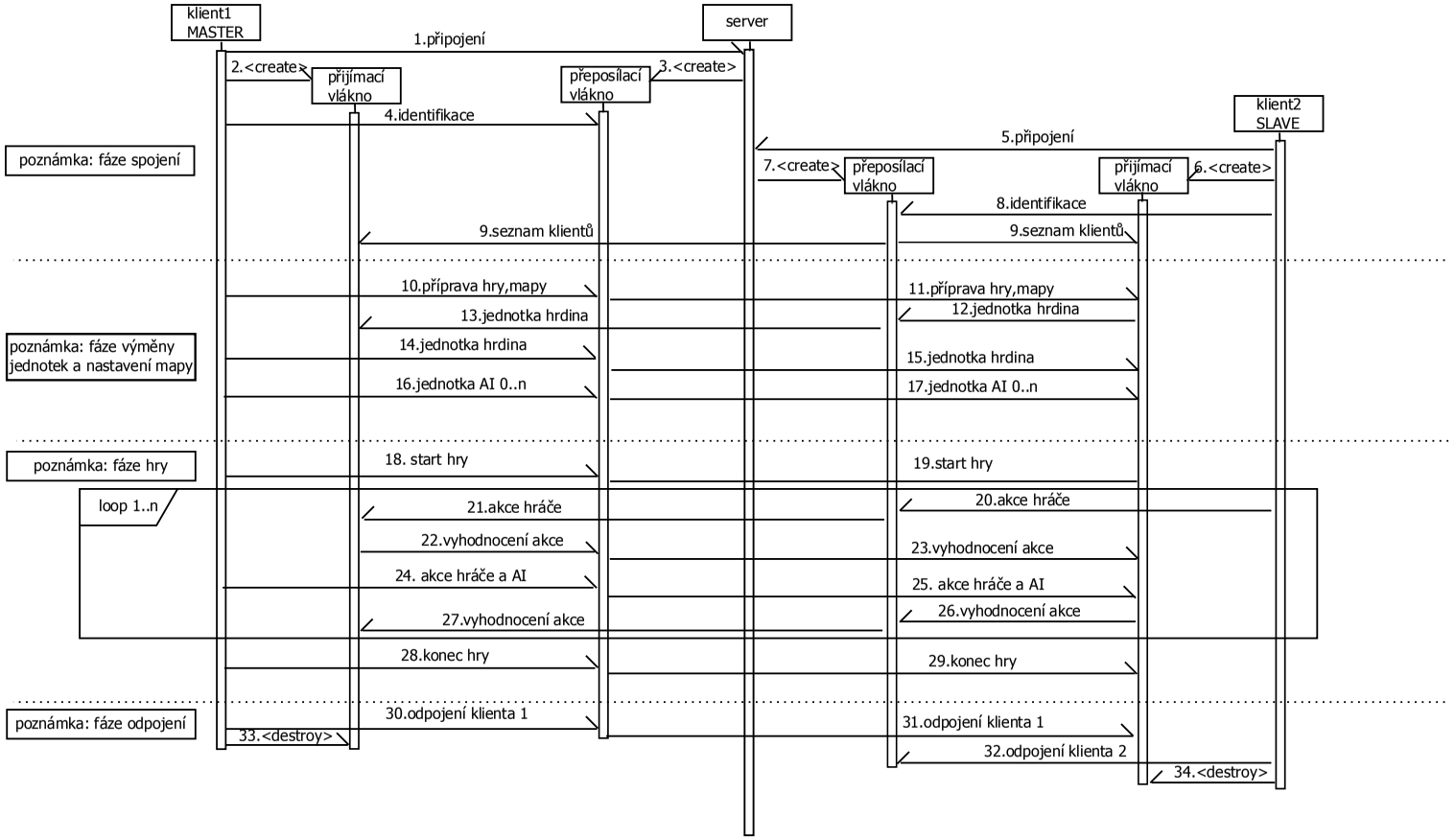
Pro komunikaci jsem se rozhodl prozatím využít potvrzovanou komunikaci pomocí TCP/IP spojení, protože u UDP je nutné řešit dodatečné kontroly, zda všechna data, které jsou odeslány dorazí v pořádku na druhou stranu. Tímto řešením však zvětšuji nároky na síťovou komunikaci, jelikož všechny pakety takto poslané vyžadují potvrzování. To bude znamenat větší odezvy komunikace, ale vyšší bezpečnost přenosu je pro mě důležitější. Navíc je využití TCP/IP jednodušší.

8.2.4 Rozložení komunikace

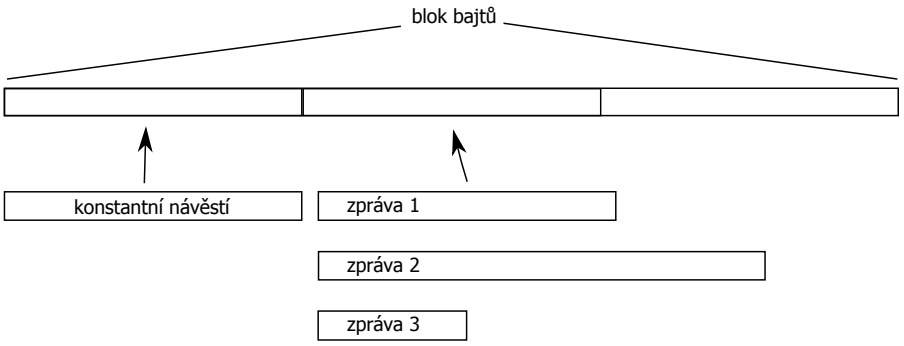
Dobré rozložení úkolů serveru a klientů umožní podstatně jednodušší komunikaci. Pro náš účel jsem zvolil co nejjednodušší server s minimální logikou a veškeré řízení hry přenechat klientům. Server bude proto řešit pouze přihlašování a odhlašování klientů. Každému přidělí volný socket a číslo, pod kterým budou ve hře identifikováni a podle toho i adresováni. Po přihlášení klienta informuje server vždy všechny přihlášené klienty o novém klientovi, jeho jméně a identifikačním číslu pod kterým se připojil. Při odpojení zašle všem klientům informace o odpojení konkrétního klienta. Zprávy bude přeposílat podle stanovených podmínek buď všem klientům, nebo jen jednomu určitému. Sekvenční diagram viz obrázek 13 zobrazuje postup jakým při hře bude probíhat výměna informací. Zprávy v sekvenčním diagramu jsou asynchronní. Pokud bude tato struktura ve hře vykazovat chyby doplním ji ještě o kontrolu doručení zpráv, ale domnívám se že TCP komunikace zajistí minimálně v lokálních sítích doručení všech bloků dat, bez výpadků.

8.2.5 Formát zpráv

Pro síťovou hru jsem využil možností programovacího jazyka C++, který umožňuje kteroukoliv instanci, s konstantními typy parametrů přetypovat na pole bajtů konstantní délky a zároveň jej umožňuje přetypovat zpět na instanci kteréhokoliv datového typu se zachováním hodnot všech proměnných. Vymyslel jsem způsob jak přenést pomocí socketů libovolné data bez složité režie. Základem je, že každé data, které se v síti posílají mají stejný formát. Na začátku jsou řídící data, které rozhodují jaká instance bude následovat. A komu data zaslat. A za nimi následují data instancí s užitečnou informací. Strukturu zprávy popisuje obrázek 14.



Obrázek 13: Sekvenční diagram popisující postup přenosu dat.



Obrázek 14: Struktura přenášených dat.

8.2.6 Server - návrh implementace

Server musí být schopen přeposílání zpráv. Podle záhlaví zprávy zjistí o jakou zprávu se jedná a komu je zpráva určena. Může poslat zprávu jednotlivci, nebo všem. Server nebude generovat vlastní zprávy, dokáže pouze přeposílat a nezajímá ho obsah dat za návěstím, kromě identifikační zprávy, podle které si sestavuje tabulku klientů pro správné adresování. Kvůli využití socketů v takzvaném blokovacím módu, kdy vlákno čeká na nové zprávy, jsem se rozhodl pro každého klienta vytvořit jedno přeposílací vlákno pomocí knihovny ZThread [7]. Činnost tohoto vlákna popisuje následující výčet. Toto řešení v případě nečinnosti klienta šetří procesorový čas. Celou činnost serveru jsem shrnul v diagramu aktivit 16. Server pak neustále opakuje sled těchto činností. :

1. Přijetí zprávy.
2. Rozbalení zprávy a zjištění adresáta
3. Procházení seznamu všech připojených klientů a odeslání zprávy adresátovi.

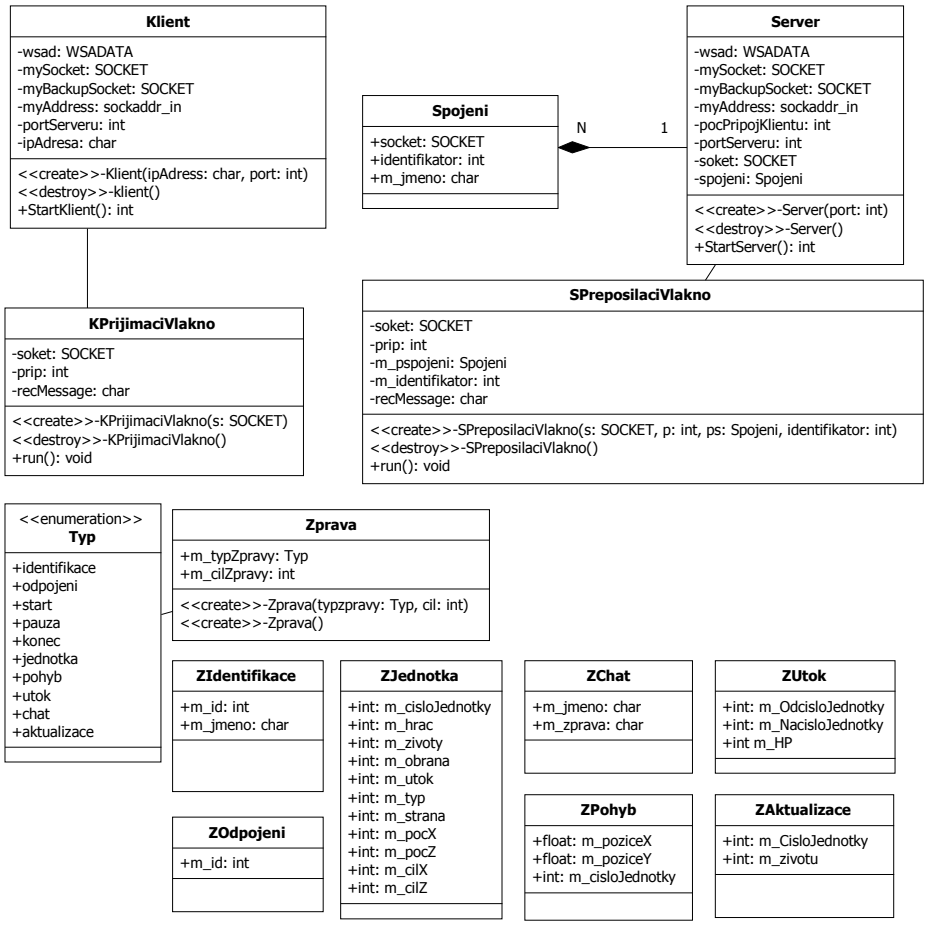
8.2.7 Klient - návrh implementace

Klient se na začátku síťové hry připojí k serveru a vytvoří si vlákno na příjem zpráv. Poté pomocí socketu začíná probíhat výměna dat a to dle aktuálních požadavků herního systému. Na konci hry se zruší přijímací vlákno a proběhne uzavření spojení. Diagram aktivit klienta na obrázku 16 je rozdělen na několik částí z důvodu, že jednotlivé části klienta se vyvolávají z různých míst herního systému.

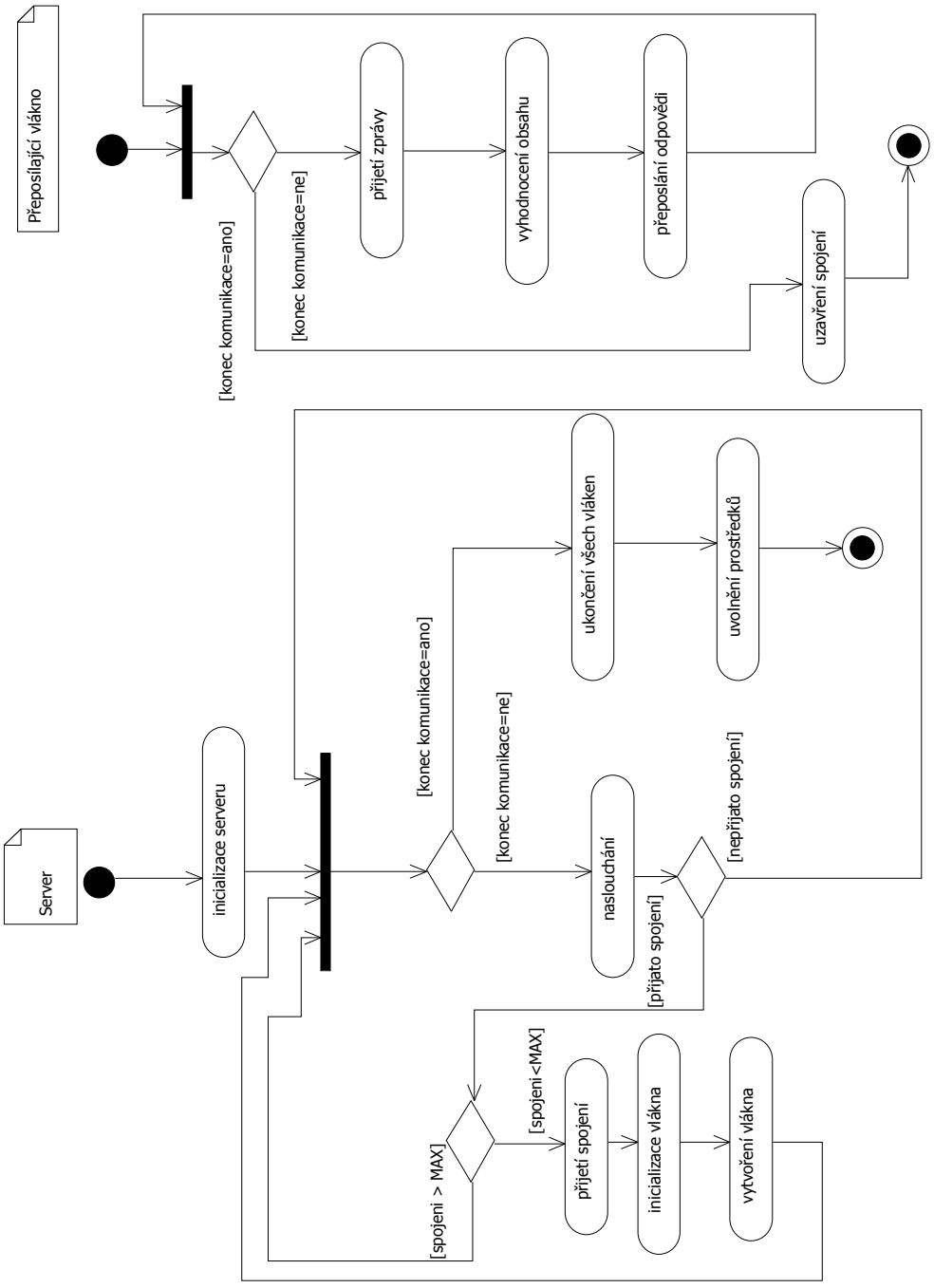
8.2.8 Implementace zpráv

Pro základní komunikaci mezi klienty byla implementována sada zpráv, jejichž výčet uvádím na třídním diagramu 15. Diagram obsahuje sadu implementovaných tříd, jejichž

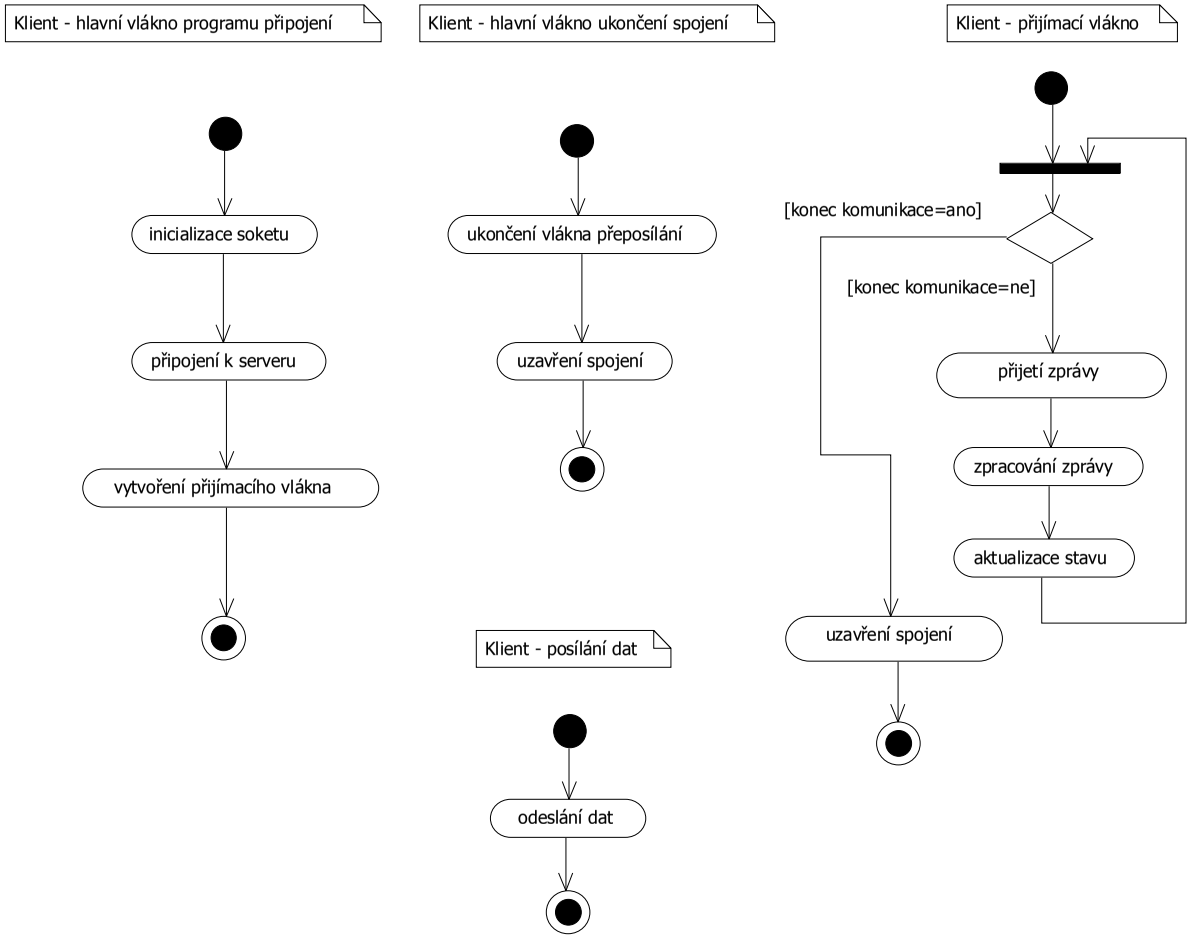
instance poslouží jako kontejnery pro přenos dat mezi klienty. Jako stejné návěstí poslouží instance Zpráva a za ní se bude řadit vždy jedna další instance. Maximální délku zprávy jsem stanovil na 256 bajtů. Později je možné formát podle potřeby zvětšovat a přidávat další kontejnery.



Obrázek 15: Třídní diagram pro server a klienty se zprávami.



Obrázek 16: Diagram aktivit serveru.



Obrázek 17: Diagram aktivit klienta.

9 Zvuk

Implementace zvukového modulu - využití jedné z dostupných knihoven pro vytváření zvukových efektů a zvukového doprovodu na pozadí hry doposud není provedena a bude součástí projektu až po dokončení herního enginu. Našim požadavkům vyhovují dvě knihovny, které mají podobnou funkcionalitu. Jednou z nich je komerční produkt FMOD. Druhým je otevřená knihovna Open AL. Obě knihovny jsem uvedl v kapitole technologie 2 Podrobnosti naleznete v referencích [8] a [9]. Implementace podpory zvuku vyžaduje mít dokončený herní systém, který bude počítat se správou zvuků ve scéně, podobně jako je tomu u 3d modelů. Do ukázkové aplikace jsem připojil knihovnu FMOD abych demonstroval možnost snadného použití knihovny. Při výběru cíle cesty přehraji krátký zvuk. Avšak plánem do budoucnosti je vložení zvuků do scény a přehrávání přizpůsobovat podle polohy posluchače. Tyto části mohou být implementovány později, neboť ozvučení není důležité pro správný chod aplikace.

9.1 Specifikace požadavků

Od zvukového doprovodu hry očekávám tyto vlastnosti, které budou později implementovány.

- Možnost přehrávat na pozadí hry hudbu a tu měnit podle událostí.
- Pro herní scénu vytvořit do stromu scény entitu zvuku s určitým okruhem a při pohledu na toto místo přehrávat všechny zvuky které se okolo vyskytují.
- Vytvořit sadu zvuků pro určité události ve hře, jako je zásah zbraní, dobytí budovy a přehrávat je, když tyto události nastanou.
- Vytvořit sadu melodií pro dotvoření atmosféry hry.
- Umožnit hráči nastavit úroveň hlasitosti zvuků a hudebního doprovodu.

10 Závěr

V bakalářské práci jsem se zabýval vývojem technologického dema pro budoucí počítačovou hru. Výsledkem mé práce je spustitelný demonstrační program(demo). Vzhled programu ukazují obrázky 19, 20, 21, 22 v příloze. Demonstrace ukazuje funkční hledání cest na mapě s překážkami, implementované pomocí A*Star algoritmu. Hledání cesty funguje pro všechny postavy ve hře, pro postavy hráčů tak i pro pozdější postavy řízené počítačem. Hledání cest je prozatím statické a neumožňuje vyhýbání se dynamickým překážkám. Zároveň demonstrační program předvádí implementaci síťové komunikace mezi několika zapnutými instancemi hry. A to tak, že přenáší postavy a jejich pohyb všem připojených hráčům. Implementace dalších činností jako jsou útoky už bude jen rozšíření klienta. Demo umožňuje zvolit myší cíl cesty a ovládat tak svou postavu v jednoduchém plochém herním světě. Díky síťové komunikaci aplikací je na všech spuštěných instancích hry je vidět synchronní činnost na stejné mapě.

Implementované části jsou však pouhým základem pro budoucí počítačovou hru. V blízké budoucnosti plánujeme propojení herního světa tvořeného Lumírem Janoškem a mých částí AI a Network pomocí jedné komponenty Game engine, která pojme mou část herní systém. Bude třeba přesně specifikovat API jednotlivých částí a navrhnout komunikaci mezi objekty. A postupně doplnit chybějící články a vytvořit první verzi hry. Hru plánujeme testovat na mém počítači, na kterém poběží server a www stránky projektu odkud si budou moci zájemci z řad studentů na kolejích a v rámci školní sítě hru stáhnout a testovat. Je pravděpodobné že bude v budoucnu tato kompletní počítačová hra předmětem mé diplomové práce.

V bakalářské práci jsem se snažil dodržet sled operací podle vodopádového modelu vývoje aplikací. Zaměřením na některé části však nebylo možné postihnout všechny problémy spjaté z kompletní hrou. Proto zůstaly pro budoucí práci ještě mnohé nezodpovězené otázky, na které bude potřeba nalézt odpovědi. Vývoj počítačové hry mi dal však mnoho nových zkušeností v oblasti vývoje software, které zúročím pří pokračování a v budoucím povolání programátora/analytika.

Lubomír Březina

11 Reference

[1] Hliněný, Petr, *Diskrétní Matematika*, ČR: VŠB,2006, 456-533 DIM

[2] Demel,Jiří, *Grafy a jejich aplikace*, ČR: Academia, 2002, ISBN 80-200-0990-6.

[3] Millington, Ian, *Artificial Inteligence for Games*, San Francisco: Elseiver Inc., 2006, ISBN 13:978-0-12-497782-2

[4] Walnum, Clayton *Programujeme grafiku v Microsoft Direct3D*, ČR: Computer Press Brno, 2004, ISBN 80-251-0136-3

[5] Microsoft Corporation, *MSDN - DirectX SDK*
http://msdn.microsoft.com/en-us/library/ee416796(v=VS.85).aspx

[6] Microsoft Corporation, *MSDN - Windows Sockets 2*
http://msdn.microsoft.com/en-us/library/ms740673(VS.85).aspx

[7] Eric Crahen *ZThread - stránky projektu*
http://zthread.sourceforge.net/.

[8] Firelight Technologies *FMOD - stránky projektu*
http://www.fmod.org/index.php/fmod.

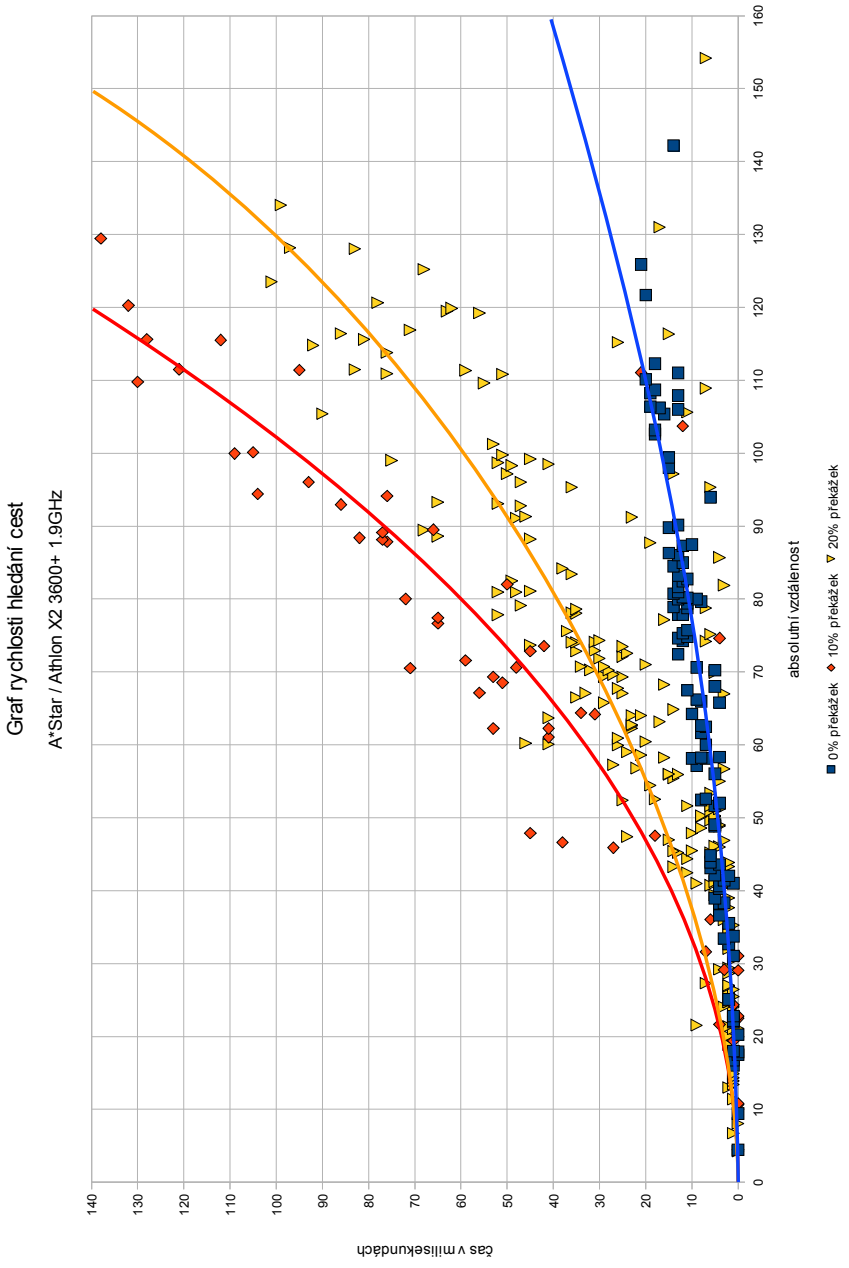
[9] Creative Labs *OpenAl - stránky projektu*
http://connect.creativelabs.com/openal/default.aspx.

[10] PATHEngine,Francie *PATHEngine - stránky knihovny*
http://www.pathengine.com/index.php.

[11] Lee Thomason, San Francisco *MicroPather - stránky projektu*
http://www.grinninglizard.com/MicroPather/.

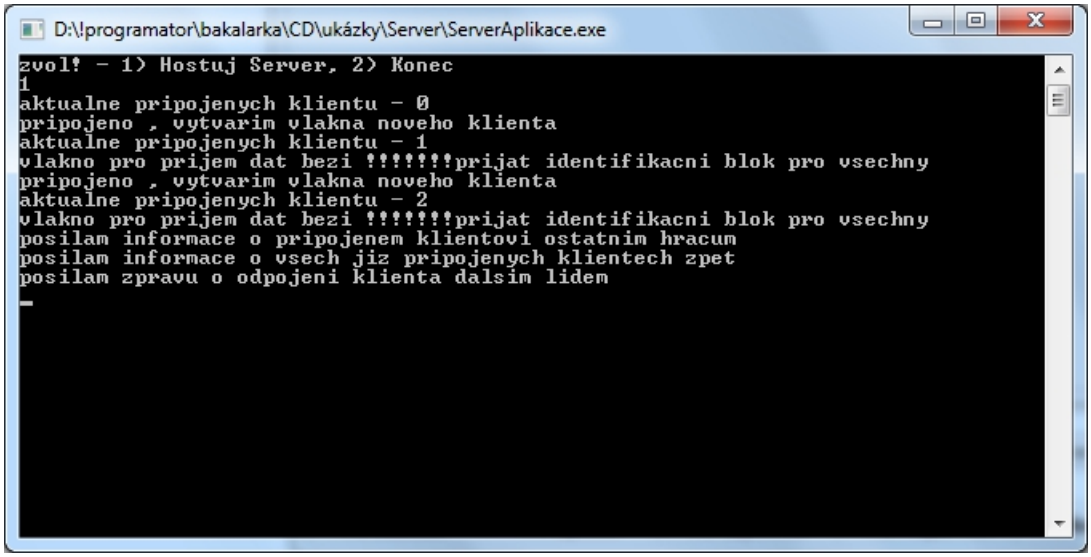
[12] Autodesk, Inc *Autodesk Kynapse - stránky firmy Autodesk*
http://usa.autodesk.com/.

A Graf výkonu

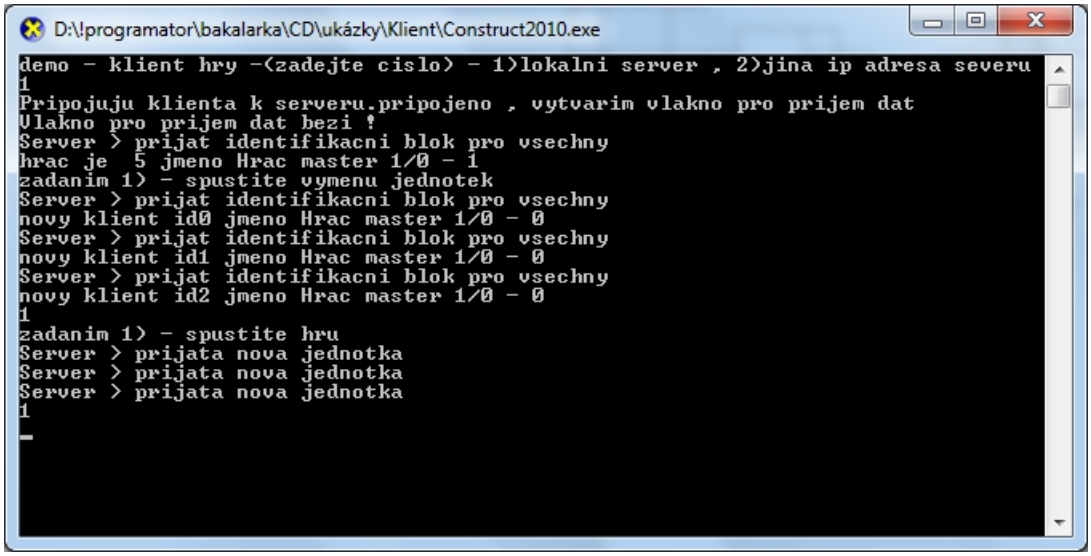


Obrázek 18: Graf výkonu hledání cest pro různé mapy.

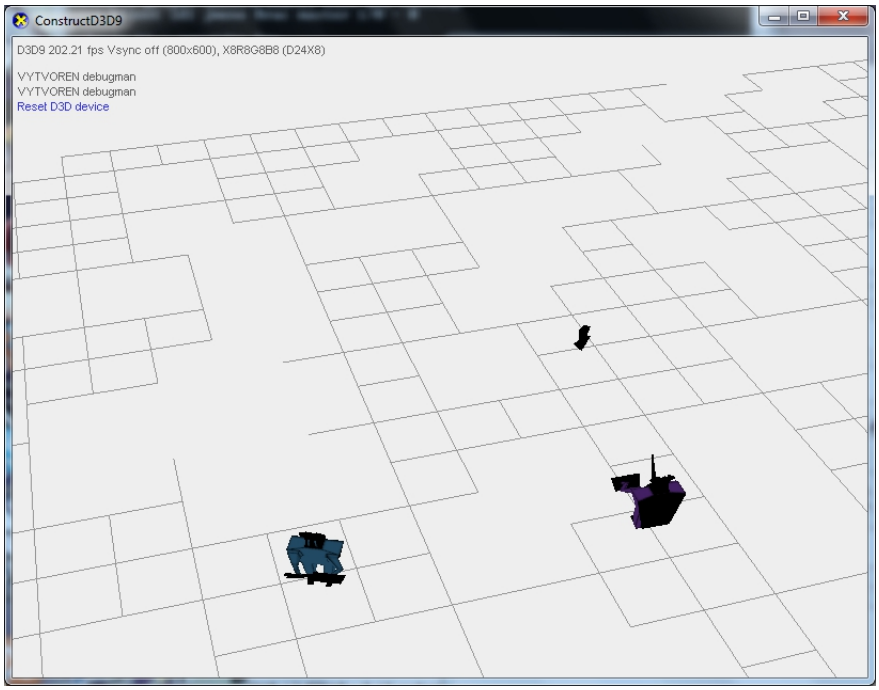
B Ukázky demonstračního programu



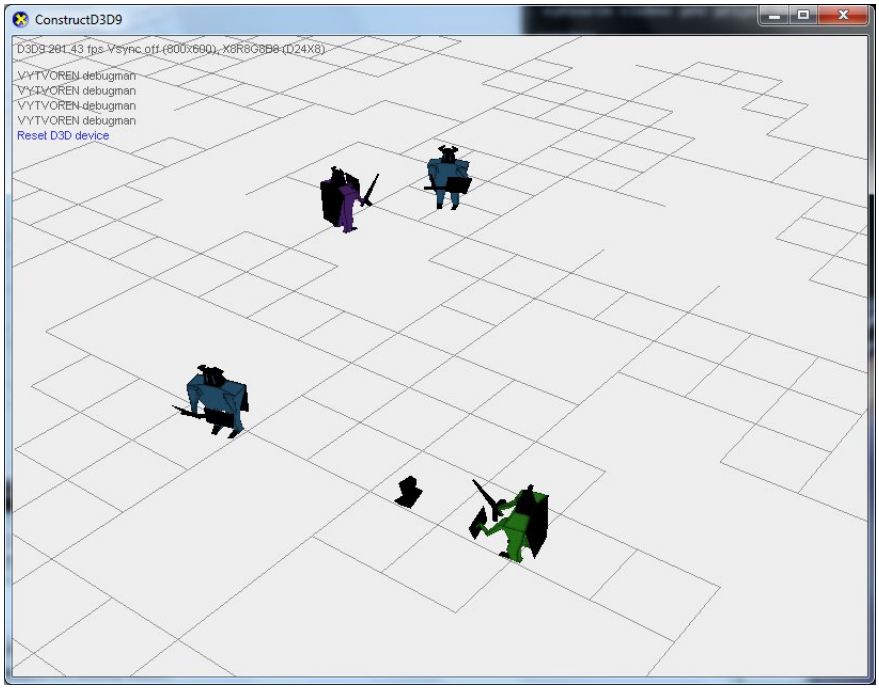
Obrázek 19: Ukázka výpisu serveru



Obrázek 20: Ukázka výpisu klienta



Obrázek 21: Ukázka hry dvou hráčů



Obrázek 22: Ukázka hry čtyř hráčů

C Obsah CD

- Text této práce ve formátu pdf.
- Zdrojové kódy programu.
- Zkompilovaný a spustitelný program serveru.
- Zkompilovaný a spustitelný program klienta s ukázkou výměny dat a hledání cest.
- Textury a modely využívané v aplikaci.
- Microsoft DirectX Redistributable.
- Návod NAVOD.TXT popisující spuštění obou programů